

# Red Hat Enterprise Linux 6 Administration des Logical Volume Manager

---

LVM-Administratorhandbuch
Ausgabe 1

Landmann

# Red Hat Enterprise Linux 6 Administration des Logical Volume Manager

---

## LVM-Administratorhandbuch
## Ausgabe 1

Landmann
rlandmann@redhat.com

## Rechtlicher Hinweis



## Zusammenfassung

Dieses Buch beschreibt den LVM Logical Volume Manager und umfasst Informationen für das Ausführen von LVM in einer Cluster-Umgebung.

# Inhaltsverzeichnis

# Einführung

## 1. Über dieses Handbuch

Dieses Buch beschreibt den Logical Volume Manager (LVM) und umfasst Informationen über das Ausführen von LVM in einer Cluster-Umgebung.

## 2. Zielgruppe

Dieses Buch richtet sich an Systemadministratoren, die Systeme verwalten, auf denen das Betriebssystem Linux läuft. Kenntnisse in Red Hat Enterprise Linux 6 und der Administration des GFS2-Dateisystems werden vorausgesetzt.

## 3. Software-Versionen

**Tabelle 1. Software-Versionen**

| Software | Beschreibung |
| --- | --- |
| RHEL 6 | bezieht sich auf RHEL 6 oder höher |
| GFS2 | bezieht sich auf GFS2 für RHEL 6 und höher |

## 4. Verwandte Dokumentation

Werfen Sie einen Blick auf die folgenden Quellen für weitere Informationen zur Verwendung von Red Hat Enterprise Linux:

- *Installationshandbuch* — Liefert Informationen bezüglich der Installation von Red Hat Enterprise Linux 6.
- *Deployment-Handbuch* — Liefert Informationen zum Einsatz, der Konfiguration und der Administration von Red Hat Enterprise Linux 6.
- *Handbuch zur Speicheradministration* — Liefert Informationen über die effiziente Verwaltung von Speichergeräten und Dateisystemen auf Red Hat Enterprise Linux 6.

Werfen Sie für weitere Informationen über das Hochverfügbarkeits-Add-On und das Resilient-Storage-Add-On für Red Hat Enterprise Linux 6 einen Blick auf die folgenden Quellen:

- *Überblick über das Hochverfügbarkeits-Add-On* — Liefert einen umfassenden Überblick über das Hochverfügbarkeits-Add-On.
- *Cluster-Administration* — Liefert Informationen zur Installation, Konfiguration und Verwaltung des Red Hat Hochverfügbarkeits-Add-Ons.
- *Global File System: Konfiguration und Administration* — Liefert Informationen zur Installation, Konfiguration und Wartung des Red Hat GFS2 (Red Hat Global File System 2), das Teil des Resilient-Storage-Add-On ist.
- *DM-Multipath* — Liefert Informationen zur Verwendung des Device-Mapper Multipath-Features von Red Hat Enterprise Linux 6.
- *Lastverteilungs-Administration* — Liefert Informationen zur Konfiguration von Hochleistungssystemen und -diensten mit dem Red Hat Lastverteilungs-Add-On, einer Gruppe integrierter Software-Komponenten, die Linux Virtual Server (LVS) bereitstellen, um IP-Lasten über eine Gruppe realer Server zu verteilen.

- *Release Notes* — Liefert Informationen über die jeweils aktuelle Release der Red Hat Produkte.

Dokumentation zum Hochverfügbarkeits-Add-On und andere Red Hat Dokumente stehen als HTML-, PDF-, und RPM-Versionen auf der Red Hat Enterprise Linux Dokumentations-CD sowie online unter http://www.redhat.com/docs/ zur Verfügung.

## 5. Wir freuen uns auf Ihr Feedback!

Falls Sie einen Fehler in diesem Handbuch finden oder eine Idee haben, wie dieses verbessert werden könnte, freuen wir uns über Ihr Feedback! Bitte reichen Sie einen Fehlerbericht in Bugzilla (http://bugzilla.redhat.com/) für das Produkt **Red Hat Enterprise Linux 6** und die Komponente **doc-Logical_Volume_Manager** ein. Vergewissern Sie sich beim Einreichen eines Fehlerberichts, dass Sie die Kennung des Handbuchs mit angeben: **Logical_Volume_Manager_Administration(EN)-6 (2011-05-19-15:20)**.

Falls Sie uns einen Vorschlag zur Verbesserung der Dokumentation senden möchten, sollten Sie hierzu möglichst genaue Angaben machen. Wenn Sie einen Fehler gefunden haben, geben Sie bitte die Nummer des Abschnitts und einen Ausschnitt des Textes an, damit wir diesen leicht finden können.

## 6. Dokumentkonventionen

Dieses Handbuch verwendet mehrere Konventionen, um bestimmte Wörter und Sätze hervorzuheben und Aufmerksamkeit auf bestimmte Informationen zu lenken.

In PDF- und Papierausgaben verwendet dieses Handbuch Schriftbilder des Liberation-Fonts-Sets. Das Liberation-Fonts-Set wird auch für HTML-Ausgaben verwendet, falls es auf Ihrem System installiert ist. Falls nicht, werden alternative, aber äquivalente Schriftbilder angezeigt. Beachten Sie: Red Hat Enterprise Linux 5 und die nachfolgende Versionen beinhalten das Liberation-Fonts-Set standardmäßig.

### 6.1. Typografische Konventionen

Es werden vier typografische Konventionen verwendet, um die Aufmerksamkeit auf bestimmte Wörter und Sätze zu lenken. Diese Konventionen und die Umstände, unter denen sie auftreten, sind folgende:

**Nichtproportional Fett**

Dies wird verwendet, um Systemeingaben hervorzuheben, einschließlich Shell-Befehle, Dateinamen und -pfade. Es wird ebenfalls zum Hervorheben von Tasten und Tastenkombinationen verwendet. Zum Beispiel:

> Um den Inhalt der Datei **my_next_bestselling_novel** in Ihrem aktuellen Arbeitsverzeichnis zu sehen, geben Sie den Befehl **cat my_next_bestselling_novel** in den Shell-Prompt ein und drücken Sie **Enter**, um den Befehl auszuführen.

Das oben aufgeführte Beispiel beinhaltet einen Dateinamen, einen Shell-Befehl und eine Taste. Alle werden nichtproportional fett dargestellt und alle können, dank des Kontextes, leicht unterschieden werden.

Tastenkombinationen unterscheiden sich von einzelnen Tasten durch das Pluszeichen, das die einzelnen Teile einer Tastenkombination miteinander verbindet. Zum Beispiel:

> Drücken Sie **Enter**, um den Befehl auszuführen.
>
> Drücken Sie **Strg**+**Alt**+**F2**, um zu einem virtuellen Terminal zu wechseln.

Das erste Beispiel hebt die zu drückende Taste hervor. Das zweite Beispiel hebt eine Tastenkombination hervor: eine Gruppe von drei Tasten, die gleichzeitig gedrückt werden müssen.

Falls Quellcode diskutiert wird, werden Klassennamen, Methoden, Funktionen, Variablennamen und Rückgabewerte, die innerhalb eines Abschnitts erwähnt werden, wie oben gezeigt **`nichtproportional fett`** dargestellt. Zum Beispiel:

> Zu dateiverwandten Klassen zählen **`filesystem`** für Dateisysteme, **`file`** für Dateien und **`dir`** für Verzeichnisse. Jede Klasse hat ihren eigenen Satz an Berechtigungen.

**Proportional Fett**

Dies kennzeichnet Wörter oder Sätze, die auf einem System vorkommen, einschließlich Applikationsnamen, Text in Dialogfeldern, beschriftete Schaltflächen, Bezeichnungen für Auswahlkästchen und Radio-Buttons, Überschriften von Menüs und Untermenüs. Zum Beispiel:

> Wählen Sie **System** → **Einstellungen** → **Maus** in der Hauptmenüleiste aus, um die **Mauseinstellungen** zu öffnen. Wählen Sie im Reiter **Tasten** auf das Auswahlkästchen **Mit links bediente Maus** und anschließend auf **Schließen**, um die primäre Maustaste von der linken auf die rechte Seite zu ändern (d.h., um die Maus auf Linkshänder anzupassen).
>
> Um ein Sonderzeichen in eine **gedit**-Datei einzufügen, wählen Sie **Anwendungen** → **Zubehör** → **Zeichentabelle** aus der Hauptmenüleiste. Wählen Sie als Nächstes **Suchen** → **Suchen** aus der Menüleiste der **Zeichentabelle**, geben Sie im Feld **Suchbegriff** den Namen des Zeichens ein und klicken Sie auf **Weitersuchen**. Das gesuchte Zeichen wird daraufhin in der **Zeichentabelle** hervorgehoben. Doppelklicken Sie auf dieses hervorgehobene Zeichen, um es in das Feld **Zu kopierender Text** zu übernehmen und klicken Sie anschließend auf die Schaltfläche **Kopieren**. Gehen Sie nun zurück in Ihr Dokument und wählen Sie **Bearbeiten** → **Einfügen** aus der **gedit**-Menüleiste.

Der oben aufgeführte Text enthält Applikationsnamen, systemweite Menünamen und -elemente, applikationsspezifische Menünamen sowie Schaltflächen und Text innerhalb einer grafischen Oberfläche. Alle werden proportional fett dargestellt und sind anhand des Kontextes unterscheidbar.

***`Nichtproportional Fett Kursiv`*** oder ***Proportional Fett Kursiv***

Sowohl bei nichtproportional fett als auch bei proportional fett weist ein zusätzlicher Kursivdruck auf einen ersetzbaren oder variablen Text hin. Kursivdruck kennzeichnet Text, der nicht wörtlich eingeben wird, oder angezeigten Text, der sich abhängig von den gegebenen Umständen unterscheiden kann. Zum Beispiel:

> Um sich mit einer Remote-Maschine via SSH zu verbinden, geben Sie an einem Shell-Prompt **`ssh`** ***`username@domain.name`*** ein. Falls die Remote-Maschine **`example.com`** ist und Ihr Benutzername auf dieser Maschine John lautet, geben Sie also **`ssh john@example.com`** ein.
>
> Der Befehl **`mount -o remount`** ***`file-system`*** hängt das angegebene Dateisystem wieder ein. Um beispielsweise das **`/home`**-Dateisystem wieder einzuhängen, verwenden Sie den Befehl **`mount -o remount /home`**.
>
> Um die Version des derzeit installierten Pakets zu sehen, verwenden Sie den Befehl **`rpm -q`** ***`package`***. Die Ausgabe sieht wie folgt aus: ***`package-version-release`***.

Beachten Sie die kursiv dargestellten Begriffe oben — username, domain.name, file-system, package,

version und release. Jedes Wort ist ein Platzhalter entweder für Text, den Sie für einen Befehl eingeben, oder für Text, der vom System angezeigt wird.

Neben der Standardbenutzung für die Darstellung des Titels eines Werks zeigt der Kursivdruck auch die erstmalige Verwendung eines neuen und wichtigen Begriffs an. Zum Beispiel:

> Publican ist ein *DocBook* Publishing-System.

## 6.2. Konventionen für Seitenansprachen

Ausgaben des Terminals und Auszüge aus dem Quellcode werden visuell vom umliegenden Text hervorgehoben durch sogenannte Seitenansprachen (auch Pull-Quotes genannt).

Eine an das Terminal gesendete Ausgabe wird in den Schrifttyp **`nichtproportional Roman`** gesetzt und wie folgt dargestellt:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Auszüge aus dem Quellcode werden ebenfalls in den Schrifttyp **`nichtproportional Roman`** gesetzt, doch wird zusätztlich noch die Syntax hervorgehoben:

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                    struct kvm_assigned_pci_dev *assigned_dev)
{
          int r = 0;
          struct kvm_assigned_dev_kernel *match;

          mutex_lock(&kvm->lock);

          match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                        assigned_dev->assigned_dev_id);
          if (!match) {
                    printk(KERN_INFO "%s: device hasn't been assigned before, "
                      "so cannot be deassigned\n", __func__);
                    r = -EINVAL;
                    goto out;
          }

          kvm_deassign_device(kvm, match);

          kvm_free_assigned_device(kvm, match);

out:
          mutex_unlock(&kvm->lock);
          return r;
}
```

## 6.3. Anmerkungen und Warnungen

Zu guter Letzt verwenden wir drei visuelle Stile, um die Aufmerksamkeit auf Informationen zu lenken, die andernfalls vielleicht übersehen werden könnten.



### Anmerkung

Eine Anmerkung ist ein Tipp, ein abgekürztes Verfahren oder ein alternativer Ansatz für die vorliegende Aufgabe. Das Ignorieren von Anmerkungen sollte keine negativen Auswirkungen haben, aber Sie verpassen so vielleicht einen Trick, der Ihnen das Leben vereinfachen könnte.



### Wichtig

Die Wichtig-Schaukästen lenken die Aufmerksamkeit auf Dinge, die sonst leicht übersehen werden können: Konfigurationsänderungen, die nur für die aktuelle Sitzung gelten oder Dienste, für die ein Neustart nötig ist, bevor eine Aktualisierung wirksam wird. Das Ignorieren von Wichtig-Schaukästen würde keinen Datenverlust verursachen, kann aber unter Umständen zu Ärgernissen und Frustration führen.



### Warnung

Eine Warnung sollte nicht ignoriert werden. Das Ignorieren von Warnungen führt mit hoher Wahrscheinlichkeit zu Datenverlust.

# Kapitel 1. Der LVM Logical Volume Manager

Dieses Kapitel liefert einen Überblick über die Features des LVM Logical Volume Manager, die in der ersten Release und in den nachfolgenden Releases von Red Hat Enterprise Linux 6 neu sind. Im Anschluss daran liefert dieses Kapitel einen umfassenden Überblick über die Komponenten des Logical Volume Manager (LVM).

## 1.1. Neue und veränderte Features

Dieser Abschnitt listet neue und veränderte Features des LVM Logical Volume Managers auf, die in der ersten Release und in den nachfolgenden Releases von Red Hat Enterprise Linux 6 enthalten sind.

### 1.1.1. Neue und veränderte Features für Red Hat Enterprise Linux 6.0

Red Hat Enterprise Linux 6.0 führt die folgenden Änderungen und Aktualisierungen an Dokumentationen und Features ein.

- Mithilfe der **`mirror_image_fault_policy`** und **`mirror_log_fault_policy`**-Parameter im **`activation`**-Abschnitt der **`lvm.conf`**-Datei können Sie definieren, wie ein gespiegelter logischer Datenträger sich bei einem Ausfall des Gerätes verhalten soll. Ist dieser Parameter auf **`remove`** gesetzt, versucht das System, das fehlerhafte Gerät zu entfernen und ohne es weiterzuarbeiten. Ist dieser Parameter auf **`allocate`** gesetzt, versucht das System, das fehlerhafte Gerät zu entfernen und Speicherplatz auf einem neuen Gerät als Ersatz für das ausgefallene Gerät zuzuweisen; diese Richtlinie verhält sich wie die **`remove`**-Richtlinie, falls kein passendes Gerät als Ersatz zugewiesen werden kann. Weitere Informationen über die Richtlinien beim Ausfall von LVM-Mirrors finden Sie in Abschnitt 4.4.3.1, „Ausfallrichtlinie für gespiegelte logische Datenträger“.
- In der Red Hat Enterprise Linux 6 Release wurde der Linux I/O-Stapel verbessert, um herstellerspezifische I/O-Grenzen zu verarbeiten. Dies ermöglicht es Speicherverwaltungs-Tools wie LVM, die Datenverteilung und den -zugriff zu optimieren. Diese Unterstützung kann deaktiviert werden, indem Sie die Standardwerte von **`data_alignment_detection`** und **`data_alignment_offset_detection`** in der **`lvm.conf`**-Datei ändern, allerdings wird davon abgeraten, diese Unterstützung zu deaktivieren.

  Informationen über Datenausrichtung in LVM sowie Informationen über das Ändern der Standardwerte für **`data_alignment_detection`** und **`data_alignment_offset_detection`** finden Sie in der Inline-Dokumentation der **`/etc/lvm/lvm.conf`**-Datei, die zudem in Anhang B, *Die LVM-Konfigurationsdateien* dokumentiert ist. Allgemeine Informationen über Unterstützung für den I/O-Stapel und I/O-Grenzen in Red Hat Enterprise Linux 6 finden Sie im *Handbuch zur Speicherverwaltung*.
- In Red Hat Enterprise Linux 6 bietet der Device Mapper direkte Unterstützung für **`udev`**-Integration. Dies synchronisiert den Device Mapper mit sämtlichen **`udev`**-Prozessen im Zusammenhang mit Device-Mapper-Geräten, einschließlich LVM-Geräten. Für Informationen über Device-Mapper-Unterstützung für den **`udev`**-Gerätemanager werfen Sie bitte einen Blick auf Abschnitt A.3, „Device-Mapper-Unterstützung für den udev-Gerätemanager“.
- In der Red Hat Enterprise Linux 6 Release können Sie den **`lvconvert --repair`**-Befehl verwenden, um ein Spiegelgerät nach Plattenausfall zu reparieren. Dadurch wird das Spiegelgerät wieder auf einen konsistenten Zustand gebracht. Informationen über den **`lvconvert --repair`**-Befehl finden Sie in Abschnitt 4.4.3.3, „Gespiegelte Datenträger reparieren“.
- Ab der Red Hat Enterprise Linux 6 Release können Sie die **`--merge`**-Option des **`lvconvert`**-Befehls verwenden, um einen Snapshot wieder mit dem ursprünglichen Datenträger zusammenzuführen. Informationen über das Zusammenführen von Snapshots finden Sie in Abschnitt 4.4.5, „Snapshot-Datenträger zusammenführen“.
- Ab der Red Hat Enterprise Linux 6 Release können Sie den **`--splitmirrors`**-Parameter des

**lvconvert**-Befehls verwenden, um ein redundantes Image eines gespiegelten logischen Datenträgers abzutrennen und einen neuen logischen Datenträger zu bilden. Weitere Informationen über die Verwendung dieser Option finden Sie in Abschnitt 4.4.3.2, „Abtrennen eines redundanten Images von einem gespiegelten logischen Datenträger“.

- Sie können nun ein Mirror-Protokoll für einen gespiegelten logischen Datenträger anlegen, das selbst wiederum gespiegelt ist, indem Sie beim Erstellen des gespiegelten logischen Datenträgers den **--mirrorlog mirrored**-Parameter des **lvcreate**-Befehls verwenden. Weitere Informationen zur Verwendung dieser Option finden Sie in Abschnitt 4.4.3, „Gespiegelte Datenträger erstellen“.

### 1.1.2. Neue und veränderte Features für Red Hat Enterprise Linux 6.1

Red Hat Enterprise Linux 6.1 führt die folgenden Änderungen und Aktualisierungen an Dokumentationen und Features ein.

- Die Red Hat Enterprise Linux 6.1 Release unterstützt die Erstellung eines logischen Snapshot-Datenträgers von gespiegelten logischen Datenträgern. Sie erstellen einen Snapshot eines gespiegelten Datenträgers genau so, wie Sie auch einen Snapshot eines linearen oder gestripten logischen Datenträgers erstellen würden. Weitere Informationen zur Erstellung von Snapshot-Datenträgern finden Sie in Abschnitt 4.4.4, „Snapshot-Datenträger erstellen“.
- Beim Erweitern eines LVM-Datenträgers können Sie die **--alloc cling**-Option des **lvextend**-Befehls verwenden, um die **cling**-Zuweisungsrichtlinie zu spezifizieren. Diese Richtlinie wählt Speicherplatz auf denselben physischen Datenträgern, auf denen sich das letzte Segment des vorhandenen logischen Datenträgers befindet. Falls die physischen Datenträger nicht ausreichend Platz bieten und eine Liste mit Tags in der **lvm.conf**-Datei definiert ist, überprüft LVM, ob diese Tags mit den physischen Datenträgern verknüpft sind und versucht, diese physischen Datenträger-Tags zwischen den vorhandenen und den neuen Extents abzugleichen.

  Weitere Informationen über das Erweitern von gespiegelten LVM-Datenträgern mit der **--alloc cling**-Option des **lvextend**-Befehls finden Sie unter Abschnitt 4.4.12.2, „Erweitern eines logischen Datenträgers mit der **cling**-Zuweisungsrichtlinie“.
- Sie können nun mehrere **--addtag** und **--deltag**-Parameter innerhalb eines einzigen **pvchange**-, **vgchange**- oder **lvchange**-Befehls spezifizieren. Für Informationen über das Hinzufügen und Entfernen von Objekt-Tags, siehe Abschnitt C.1, „Hinzufügen und Entfernen von Objekt-Tags“.
- Die Liste der zulässigen Zeichen in LVM-Objekt-Tags wurde erweitert, so dass Tags nun auch die Zeichen "/", "=", "!", ":", "#", und "&" enthalten können. Für Informationen über LVM-Objekt-Tags, siehe Anhang C, *LVM Objekt-Tags*.
- Sie können nun RAID0 (Striping) und RAID1 (Mirroring) auf einem einzigen logischen Datenträger kombinieren. Wenn Sie beim Erstellen eines logischen Datenträgers gleichzeitig die Anzahl der Mirrors (**--mirrors X**) sowie die Anzahl der Stripes (**--stripes Y**) angeben, wird dadurch ein Mirror-Gerät erstellt, dessen zugrunde liegenden Geräte gestriped sind. Für weitere Informationen über das Erstellen gespiegelter logischer Datenträger, siehe Abschnitt 4.4.3, „Gespiegelte Datenträger erstellen“.
- Ab der Red Hat Enterprise Linux 6.1 Release können Sie, um eine konsistente Datensicherung eines geclusterten logischen Datenträgers durchzuführen, den Datenträger exklusiv aktivieren und dann den Snapshot erstellen. Weitere Informationen über das Aktivieren von logischen Datenträgern exklusiv auf einem Knoten finden Sie in Abschnitt 4.7, „Logische Datenträger auf einzelnen Knoten in einem Cluster aktivieren“.

## 1.2. Logische Datenträger

Das Logical Volume Management schafft eine Abstraktionsschicht über dem physischen Speicher und erlaubt Ihnen das Anlegen logischer Datenträger. Dies bietet Ihnen in vielerlei Hinsicht eine weit größere

Flexibilität als dies beim direkten Zugriff auf physischen Speicher möglich wäre. Die logischen Datenträger werden nicht begrenzt durch die Größe der physischen Platten. Zusätzlich wird die Speicherkonfiguration der Hardware der Software vorenthalten, so dass sie vergrößert und verschoben werden kann, ohne dass Applikationen gestoppt oder Dateisysteme ausgehängt werden müssten. Betriebskosten können so gesenkt werden.

Logische Datenträger bieten die folgenden Vorteile gegenüber der direkten Verwendung von physischen Speichergeräten:

- Flexible Kapazität

  Bei der Verwendung logischer Datenträger können sich Dateisysteme über mehrere Platten erstrecken, da Platten und Partitionen in einem einzelnen logischen Datenträger vereinigt werden können.
- Speicher-Pools, deren Größe verändert werden kann

  Mithilfe einfacher Software-Befehle können Sie logische Datenträger erweitern oder deren Größe verringern, ohne die zugrunde liegenden Plattengeräte neu zu formatieren oder neu zu partitionieren.
- Online-Datenumzug

  Um neuere, schnellere oder robustere Speicher-Subsysteme einzusetzen, können Sie Daten verschieben, während Ihr System aktiv ist. Daten können auf Platten neu angeordnet werden, während die Platten verwendet werden. So können Sie eine Platte leeren, die im laufenden Betrieb ausgetauscht werden kann, bevor Sie diese entfernen.
- Bequeme Benennung von Geräten

  Logische Speicherdatenträger können in benutzerdefinierten Gruppen verwaltet werden, die Sie nach Belieben benennen können.
- Platten-Striping

  Sie können einen logischen Datenträger erstellen, der Daten zwischen zwei oder mehreren Platten verteilt (striped). Dies kann den Datendurchsatz drastisch erhöhen.
- Datenträger zur Spiegelung

  Logische Datenträger bieten einen bequemen Weg zur Konfiguration eines Mirrors für Ihre Daten.
- Datenträger-Snapshots

  Indem Sie logische Datenträger verwenden, können Sie Geräte-Snapshots für konsistente Sicherungen aufnehmen oder die Auswirkungen von Änderungen testen, ohne die tatsächlichen Daten zu gefährden.

Die Implementierung dieser Features in LVM wird im weiteren Verlauf dieses Dokuments beschrieben.

## 1.3. Überblick über die LVM-Architektur

Im Rahmen des Releases des Linux Betriebssystems RHEL 4 wurde der ursprüngliche LVM1 Logical Volume Manager durch LVM2 ersetzt, welches ein allgemeineres Kernel-Framework besitzt, als LVM1. LVM2 bietet die folgenden Verbesserungen gegenüber LVM1:

- flexible Kapazität
- effizienterer Metadatenspeicher
- besseres Wiederherstellungsformat
- neues ASCII-Metadatenformat
- atomische Änderungen an Metadaten
- redundante Kopien von Metadaten

LVM2 ist abwärtskompatibel zu LVM1, mit Ausnahme der Unterstützung von Snapshot und Cluster. Sie

können eine Datenträgergruppe mit dem Befehl **vgconvert** vom LVM1-Format in das LVM2-Format konvertieren. Werfen Sie einen Blick auf die Handbuchseite (8) von **vgconvert** für Informationen zur Konvertierung des Formats von LVM-Metadaten.

Die zugrunde liegende physische Speichereinheit eines logischen LVM-Datenträgers ist ein Blockgerät, wie z.B. eine Partition oder eine gesamte Platte. Dieses Gerät wird als ein *physischer Datenträger* (kurz PV für "physical volume") initialisiert.

Um einen logischen LVM-Datenträger zu erstellen, werden die physischen Datenträger in einer *Datenträgergruppe* (kurz VG für "volume group") kombiniert. Dies erstellt einen Pool aus Plattenplatz, aus dem logische LVM-Datenträger (kurz LV für "logical volume") zugewiesen werden können. Dieser Prozess entspricht der Vorgehensweise beim Partitionieren von Festplatten. Ein logischer Datenträger wird von Dateisystemen und Applikationen (wie z.B. Datenbanken) verwendet.

Abbildung 1.1, „Komponenten von logischen LVM-Datenträgern“ zeigt die Komponenten eines einfachen logischen LVM-Datenträgers:

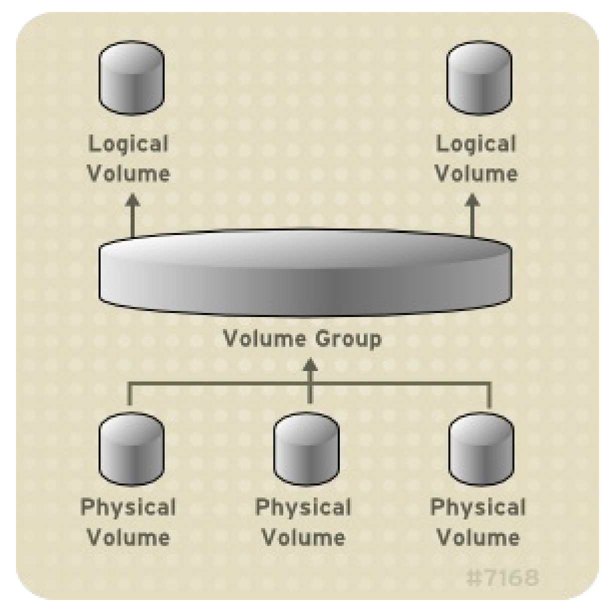



**Abbildung 1.1. Komponenten von logischen LVM-Datenträgern**

Werfen Sie einen Blick auf Kapitel 2, *LVM-Komponenten* für detaillierte Informationen zu den Komponenten eines logischen LVM-Datenträgers.

## 1.4. Der Clustered Logical Volume Manager (CLVM)

Der Clustered Logical Volume Manager (CLVM) ist eine Reihe von Cluster-Erweiterungen für LVM. Diese Erweiterungen ermöglichen einem Computer-Cluster die Verwaltung von gemeinsam genutzten Speicher (z.B. auf einem SAN) unter Verwendung von LVM. CLVM ist Teil des Resilient Storage Add-On.

Ob Sie CLVM einsetzen sollten, hängt von Ihren Systemanforderungen ab:

- Erfordert nur ein Knoten Ihres Systems Zugriff auf den Speicher, den Sie als logische Datenträger konfigurieren, so können Sie LVM ohne die CLVM-Erweiterungen einsetzen und die mit diesem Knoten erzeugten logischen Datenträger sind alle lokal für diesen Knoten.
- Falls Sie ein geclustertes System zur Ausfallsicherheit verwenden, in dem zu jeder Zeit nur ein Knoten aktiv ist, der auf den Speicher zugreift, sollten Sie High Availability Logical Volume Management Agents (HA-LVM) einsetzen. Weitere Informationen über HA-LVM finden Sie im

Handbuch *Konfiguration und Verwaltung eines Red Hat Cluster*.

- Falls mehr als ein Knoten in Ihrem Cluster Zugriff auf den Speicher benötigt, der somit also von den aktiven Knoten gemeinsam verwendet wird, müssen Sie CLVM einsetzen. CLVM ermöglicht einem Benutzer die Konfiguration von logischen Datenträgern auf gemeinsam genutztem Speicher, indem der Zugriff auf den physischen Speicher während der Konfiguration des logischen Datenträgers gesperrt wird, und verwendet geclusterte Dienste zur Sperrung, um den gemeinsam verwendeten Speicher zu verwalten.

Um CLVM zu verwenden, muss die Hochverfügbarkeits-Add-On und Resilient Storage Add-On Software einschließlich des **`clvmd`**-Daemons laufen. Der **`clvmd`**-Daemon ist die Haupt-Cluster-Erweiterung für LVM. Der **`clmvd`**-Daemon läuft auf jedem Cluster-Computer und verteilt Aktualisierungen zu LVM-Metadaten in einem Cluster und gibt daher dieselbe Darstellung der logischen Datenträger an jeden Cluster-Computer weiter. Weitere Informationen über die Installation und die Verwaltung des Hochverfügbarkeits-Add-Ons finden Sie im Handbuch *Konfiguration und Verwaltung eines Red Hat Clusters*.

Um sicherzustellen, dass **`clvmd`** beim Booten mitgestartet wird, können Sie den **`chkconfig ... on`**-Befehl für den **`clvmd`**-Dienst ausführen, und zwar wie folgt:

```
# chkconfig clvmd on
```

Falls der **`clvmd`**-Daemon noch nicht gestartet ist, können Sie den **`service ... start`**-Befehl auf dem **`clvmd`**-Dienst ausführen, und zwar wie folgt:

```
# service clvmd start
```

Das Erstellen von logischen LVM-Datenträgern in einer geclusterten Umgebung ist identisch zum Erstellen von logischen LVM-Datenträgern auf einem einzelnen Knoten. Es gibt keinen Unterschied in den LVM-Befehlen selbst oder in der grafischen Benutzeroberfläche von LVM, wie in Kapitel 4, *LVM-Administration mit CLI-Befehlen* und Kapitel 7, *LVM-Administration mit dem LVM-GUI* beschrieben. Um die LVM-Datenträger, die Sie in einem Cluster anlegen, zu aktivieren, muss die Cluster-Infrastruktur ausgeführt werden und einsatzfähig sein.

Standardmäßig sind logische Datenträger, die mit CLVM auf gemeinsam verwendetem Speicher erstellt wurden, für alle Systeme sichtbar, die Zugriff auf diesen gemeinsam verwendeten Speicher haben. Es ist möglich, Datenträgergruppen zu erstellen, in denen alle enthaltene Speichergeräte nur für einen Knoten im Cluster sichtbar sind. Es ist ebenfalls möglich, den Status einer Datenträgergruppe von einer lokalen Datenträgergruppe zu einer geclusterten Datenträgergruppe zu ändern. Weitere Informationen diesbezüglich finden Sie in Abschnitt 4.3.2, „Datenträgergruppen in einem Cluster erstellen“ and Abschnitt 4.3.7, „Parameter einer Datenträgergruppe verändern“.



## Warnung

Wenn Sie mit CLVM Datenträgergruppen auf gemeinsam verwendetem Speicher erzeugen, stellen Sie sicher, dass alle Knoten im Cluster Zugriff auf die physischen Datenträger haben, aus denen sich die Datenträgergruppe zusammensetzt. Asymmetrische Cluster-Konfigurationen, bei denen einige Knoten Zugriff auf den Speicher haben und andere nicht, werden nicht unterstützt.

Abbildung 1.2, „CLVM-Überblick“ zeigt einen CLVM-Überblick in einem Cluster.

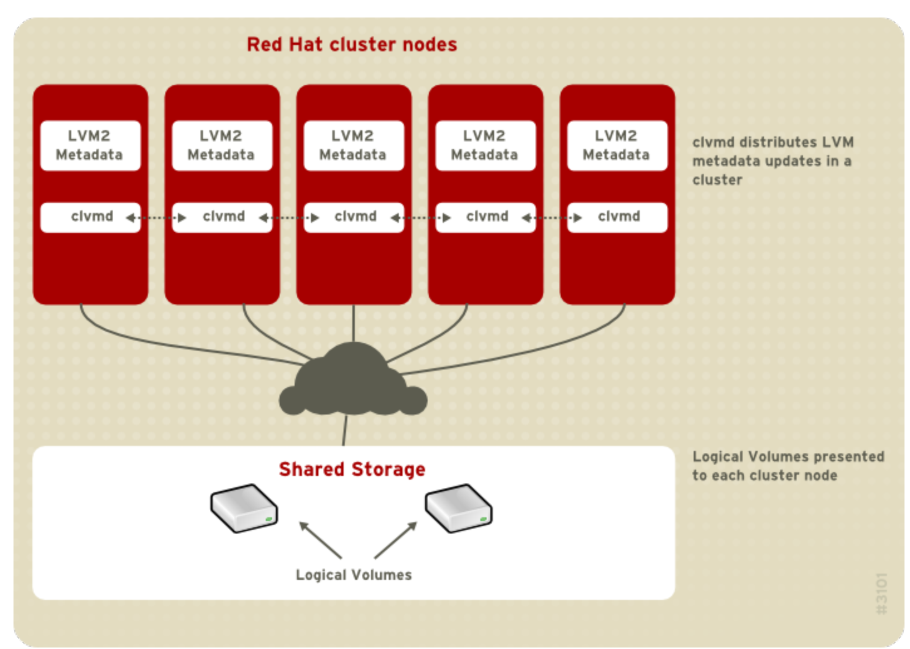



**Abbildung 1.2. CLVM-Überblick**



**Anmerkung**

CLVM erfordert Änderungen an der **`lvm.conf`**-Datei zur clusterweiten Sperrung. Sie finden Informationen über die Konfiguration der **`lvm.conf`**-Datei zur Unterstützung von Cluster-Sperrung in der **`lvm.conf`**-Datei selbst. Informationen über die**`lvm.conf`**-Datei finden Sie in Anhang B, *Die LVM-Konfigurationsdateien*.

## 1.5. Dokumentüberblick

Der Rest dieses Dokuments gliedert sich in die folgenden Kapitel:

- Kapitel 2, *LVM-Komponenten* beschreibt die Komponenten, aus denen ein logischer LVM-Datenträger besteht.
- Kapitel 3, *Überblick über die LVM-Administration* liefert einen Überblick über die grundlegenden Schritte zur Konfiguration von logischen LVM-Datenträgern, unabhängig davon, ob Sie die Kommandozeilenbefehle (CLI) für LVM oder das Graphical User Interface (GUI) für LVM verwenden.
- Kapitel 4, *LVM-Administration mit CLI-Befehlen* fasst die individuellen administrativen Aufgaben zusammen, die Sie mit den LVM CLI-Befehlen zur Erstellung und Wartung von logischen Datenträgern verwenden können.
- Kapitel 5, *Konfigurationsbeispiele für LVM* liefert eine Vielzahl an LVM-Konfigurationsbeispielen.
- Kapitel 6, *Suche und Bereinigung von LVM-Fehlern* liefert Anweisungen zur Fehlerbehebung von einer Vielzahl von LVM-Problemen.
- Kapitel 7, *LVM-Administration mit dem LVM-GUI* fasst die Funktionsweise des LVM-GUI zusammen.
- Anhang A, *Der Device-Mapper* beschreibt den Device Mapper, den LVM zur Zuordnung von logischen und physischen Datenträgern verwendet.
- Anhang B, *Die LVM-Konfigurationsdateien* beschreibt die Konfigurationsdateien von LVM.
- Anhang C, *LVM Objekt-Tags* beschreibt die LVM-Objekt-Tags und Host-Tags.
- Anhang D, *Metadaten einer LVM-Datenträgergruppe* beschreibt die Metadaten der LVM-

Datenträgergruppe und beinhaltet ein Beispielexemplar von Metadaten für eine LVM-Datenträgergruppe.

# Kapitel 2. LVM-Komponenten

Dieses Kapitel beschreibt die Komponenten eines logischen LVM-Datenträgers.

## 2.1. Physische Datenträger

Die zugrunde liegende physische Speichereinheit eines logischen LVM-Datenträgers ist ein Blockgerät, wie z.B. eine Partition oder eine gesamte Festplatte. Um das Gerät für einen logischen LVM-Datenträger zu verwenden, muss das Gerät als physischer Datenträger (PV) initialisiert werden. Durch das Initialisieren eines Blockgeräts als physischer Datenträger wird ein Label im Anfangsbereich des Geräts platziert.

Standardmäßig wird das LVM-Label im zweiten 512-Byte Sektor platziert. Sie können diesen Standardwert überschreiben, indem Sie das Label in einem der ersten 4 Sektoren platzieren. Dies ermöglicht die Co-Existenz von LVM-Datenträgern mit anderen Benutzern auf diesen Sektoren, falls notwendig.

Ein LVM-Label bietet korrekte Identifikation und Anordnung von Geräten für ein physisches Gerät, da Geräte beim Booten des Systems in jeder beliebigen Reihenfolge aktiviert werden können. Ein LVM-Label bleibt über Neustarts hinaus und innerhalb eines Clusters bestehen.

Das LVM-Label identifiziert das Gerät als einen physischen LVM-Datenträger. Es enthält eine zufällige, eindeutige Kennung (die UUID) für den physischen Datenträger. Es speichert weiterhin die Größe des Blockgeräts in Bytes und zeichnet auf, wo die LVM-Metadaten auf dem Gerät gespeichert werden.

Die LVM-Metadaten enthalten die Konfigurationsdetails der LVM-Datenträgergruppen auf Ihrem System. Standardmäßig wird ein identisches Exemplar der Metadaten in jedem Bereich der Metadaten auf jedem physischen Datenträger innerhalb der Datenträgergruppe beibehalten. LVM-Metadaten sind klein und können als ASCII gespeichert werden.

Derzeit gestattet LVM die Speicherung von 0, 1 oder 2 identischer Kopien seiner Metadaten auf jedem physischen Datenträger. Der Standardwert ist 1 Kopie. Wenn Sie die Zahl der Kopien der Metadaten auf dem physischen Datenträger einmal konfigurieren, können Sie sie zu einem späteren Zeitpunkt nicht mehr ändern. Die erste Kopie wird im Anfangsbereich des Geräts kurz hinter dem Label gespeichert. Falls eine zweite Kopie existiert, wird diese am Ende des Geräts platziert. Falls Sie aus Versehen den Bereich am Anfang Ihrer Platte überschreiben, indem Sie auf eine andere Platte schreiben, als beabsichtigt, ermöglicht Ihnen die zweite Kopie der Metadaten am Ende des Geräts die Wiederherstellung der Metadaten.

Werfen Sie einen Blick auf Anhang D, *Metadaten einer LVM-Datenträgergruppe* für detaillierte Informationen zu den LVM-Metadaten und dem Ändern von Metadaten-Parametern.

### 2.1.1. Aufbau eines physischen LVM-Datenträgers

Abbildung 2.1, „Aufbau eines physischen Datenträgers“ zeigt den Aufbau eines physischen LVM-Datenträgers. Das LVM-Label befindet sich auf dem zweiten Sektor, gefolgt vom Bereich der Metadaten und dem zur Verfügung stehenden Platz auf dem Gerät.



**Anmerkung**

Im Linux-Kernel (und im weiteren Verlauf dieses Dokuments) wird von einer Größe von 512 Bytes für Sektoren ausgegangen.

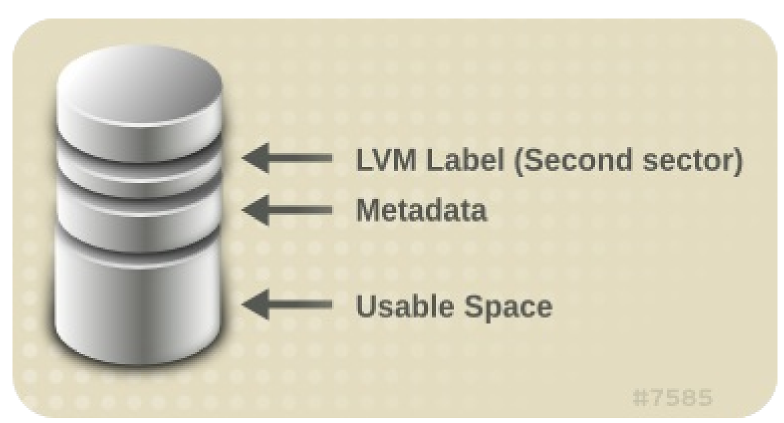



**Abbildung 2.1. Aufbau eines physischen Datenträgers**

### 2.1.2. Mehrere Partitionen auf einer Platte

LVM gestattet Ihnen, physische Datenträger aus Plattenpartitionen zu erstellen. Aus den folgenden Gründen wird allgemein empfohlen, dass Sie eine einzelne Partition erstellen, die die gesamte Platte als einen physischen Datenträger ausweist:

- Einfachere Verwaltung

  Es ist einfacher, den Überblick über Hardware in einem System zu behalten, wenn jede reale Platte nur einmal erscheint. Dies trifft besonders beim Ausfall einer Platte zu. Zusätzlich können mehrere physische Datenträger auf einer einzelnen Platte Warnungen des Kernels beim Starten bezüglich unbekannter Partitionstypen hervorrufen.

- Striping-Leistung

  LVM kann nicht ermitteln, dass sich zwei physische Datenträger auf derselben physischen Platte befinden. Falls Sie einen logischen Striped-Datenträger erstellen, während sich zwei physische Datenträger auf derselben physischen Platte befinden, könnten sich die Stripes in unterschiedlichen Partitionen auf derselben Platte befinden. Dies würde zu einem Leistungsabfall anstatt zu einer Leistungssteigerung führen.

Auch wenn es nicht empfohlen wird, gibt es ggf. bestimmte Umstände, in denen Sie eine Platte in einzelne physische LVM-Datenträger unterteilen müssen. Auf einem System mit wenigen Platten ist es beispielsweise ggf. notwendig, Daten zwischen Partition hin- und herzuverschieben, wenn Sie ein existierendes System auf LVM-Datenträger migrieren. Wenn Sie außerdem eine sehr große Platte besitzen, und aus administrativen Gründen mehr als eine Datenträgergruppe haben möchten, ist eine Partitionierung der Platte erforderlich. Wenn Sie eine Platte mit mehr als einer Partition besitzen und sich beide Partitionen in derselben Datenträgergruppe befinden, achten Sie darauf anzugeben, welche Partitionen beim Erstellen von Striped-Datenträgern in einen logischen Datenträger eingebunden werden sollen.

## 2.2. Datenträgergruppen

Physische Datenträger werden in Datenträgergruppen (VGs) zusammengefasst. Dies erstellt einen Pool aus Plattenplatz, aus dem logische Datenträger zugewiesen werden können.

Innerhalb einer Datenträgergruppe wird der für eine Zuweisung verfügbare Plattenplatz in Einheiten mit einer festen Größe unterteilt, die als Extents bezeichnet werden. Ein Extent ist die kleinste Einheit für Platz, der zugewiesen werden kann. Innerhalb eines physischen Datenträgers werden Extents als physische Extents bezeichnet.

Ein logischer Datenträger wird in logische Extents derselben Größe wie die physischen Extents unterteilt. Die Extent-Größe ist daher dieselbe für alle logischen Datenträger in der Datenträgergruppe. Die Datenträgergruppe ordnet die logischen Extents den physischen Extents zu.

## 2.3. Logische LVM-Datenträger

In LVM ist eine Datenträgergruppe in logische Datenträger unterteilt. Es gibt drei Typen von logischen LVM-Datenträger: *lineare* Datenträger, *striped* Datenträger und *mirrored* Datenträger. Diese werden in den folgenden Abschnitten beschrieben.

### 2.3.1. Lineare Datenträger

Ein linearer Datenträger vereinigt mehrere physische Datenträger in einen logischen Datenträger. Wenn Sie beispielsweise zwei 60 GB große Platten besitzen, können Sie einen logischen Datenträger mit einer Größe von 120 GB erstellen. Der physische Speicher wird verknüpft.

Beim Erstellen eines linearen Datenträgers werden eine Reihe von physischen Extents zu einem Bereich eines logischen Datenträger der Reihe nach zugewiesen. Wie beispielsweise unter Abbildung 2.2, „Extent-Zuweisung“ gezeigt, könnten die logischen Extents von 1 bis 99 zu einem physischen Datenträger und die logische Extents 100 bis 198 zu einem zweiten physischen Datenträger zugeordnet werden. Aus Sicht der Applikation existiert nur ein Gerät mit einer Größe von 198 Extents.

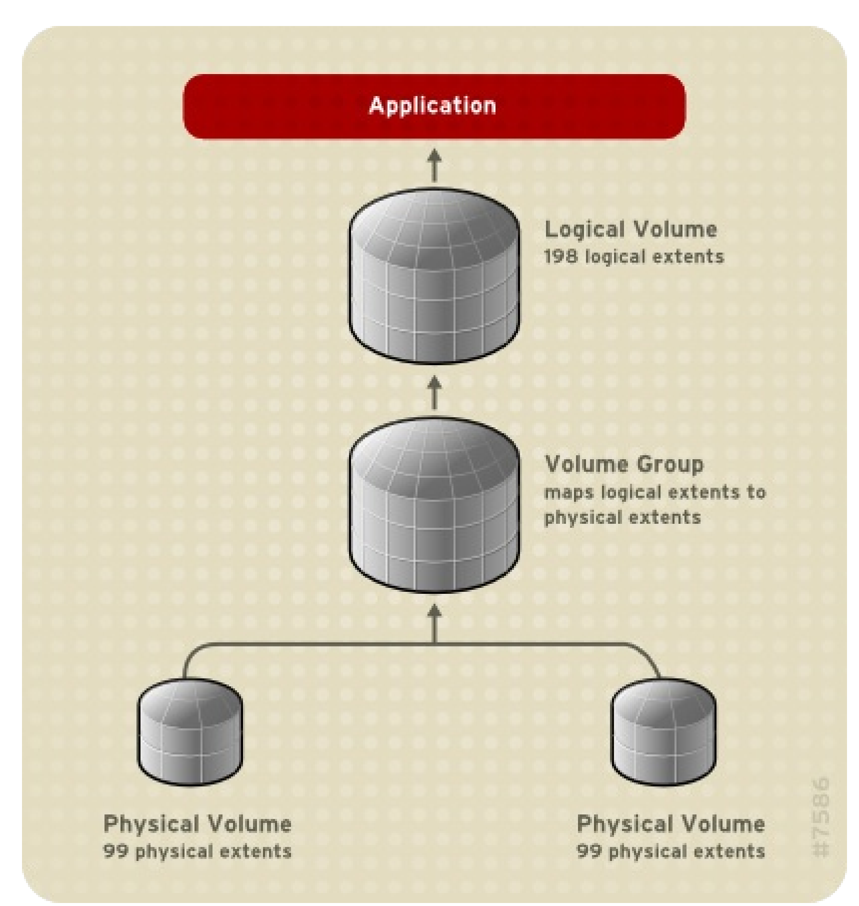



**Abbildung 2.2. Extent-Zuweisung**

Die physischen Datenträger, aus denen ein logischer Datenträger besteht, müssen nicht gleich groß sein. Abbildung 2.3, „Linearer Datenträger mit ungleichen physischen Datenträgern“ zeigt die Datenträgergruppe **VG1** mit einer Größe der physischen Extents von 4 MB. Diese Datenträgergruppe

umfasst 2 physische Datenträger namens **PV1** und **PV2**. Die physischen Datenträger sind in 4 MB Einheiten aufgeteilt, da dies die Größe der Extents ist. In diesem Beispiel ist **PV1** 100 Extents groß (400 MB) und **PV2** 200 Extents (800 MB) groß. Sie können einen linearen Datenträger mit einer Größe von 1 bis 300 Extents (4 MB bis 1200 MB) erstellen. In diesem Beispiel ist der lineare Datenträger mit der Bezeichnung **LV1** 300 Extents groß.

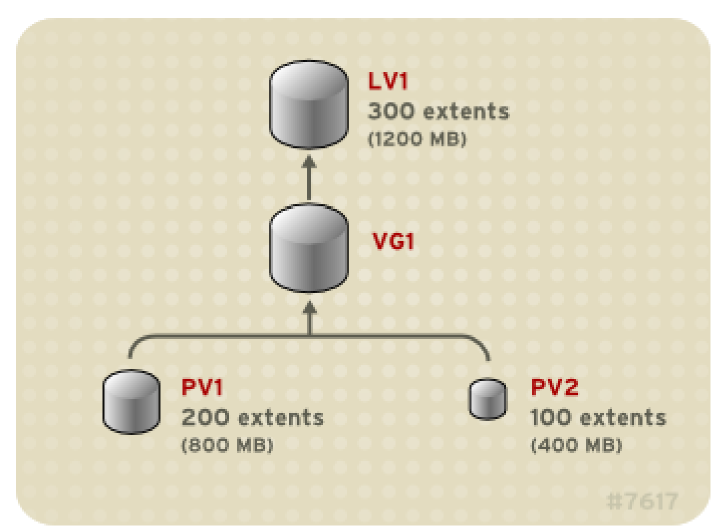



**Abbildung 2.3. Linearer Datenträger mit ungleichen physischen Datenträgern**

Sie können aus dem Pool der physischen Extents mehr als einen linearen logischen Datenträger mit einer Größe Ihrer Wahl konfigurieren. Abbildung 2.4, „Mehrere logische Datenträger“ zeigt dieselben Datenträgergruppen wie in Abbildung 2.3, „Linearer Datenträger mit ungleichen physischen Datenträgern“, jedoch wurden in diesem Fall zwei logische Datenträger aus der Datenträgergruppe erstellt: **LV1**, 250 Extents groß (1.000 MB) und **LV2**, 50 Extents groß (200 MB).

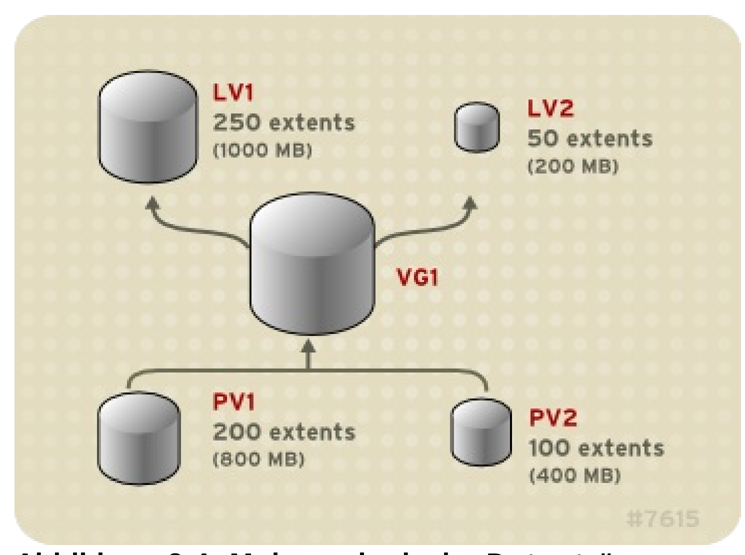



**Abbildung 2.4. Mehrere logische Datenträger**

## 2.3.2. Logische Striped-Datenträger

Beim Schreiben von Daten auf einen logischen LVM-Datenträger, verteilt das Dateisystem die Daten auf den zugrunde liegenden physischen Datenträgern. Durch die Erstellung eines logischen Striped-Datenträgers können Sie die Art und Weise kontrollieren, wie die Daten auf die physischen Datenträger

geschrieben werden. Für große sequenzielle Lese- und Schreibzugriffe kann dies die Effizienz der Daten-I/O verbessern.

Striping verbessert die Leistung, indem Daten auf eine vordefinierte Anzahl an physischen Datenträgern nach Round-Robin-Art (also reihum) geschrieben werden. Mithilfe von Striping kann I/O parallel durchgeführt werden. In einigen Situationen kann dies zu einem fast linearen Leistungszuwachs für jeden zusätzlichen physischen Datenträger in dem Stripe führen.

Die folgende Darstellung zeigt Daten, die über drei physische Datenträger hinweg gestriped werden. Die Abbildung stellt folgendes dar:

- der erste Daten-Stripe wird auf PV1 geschrieben
- der zweite Daten-Stripe wird auf PV2 geschrieben
- der dritte Daten-Stripe wird auf PV3 geschrieben
- der vierte Daten-Stripe wird auf PV1 geschrieben

In einem logischen Striped-Datenträger kann die Größe des Stripe nicht die Größe eines Extent übersteigen.

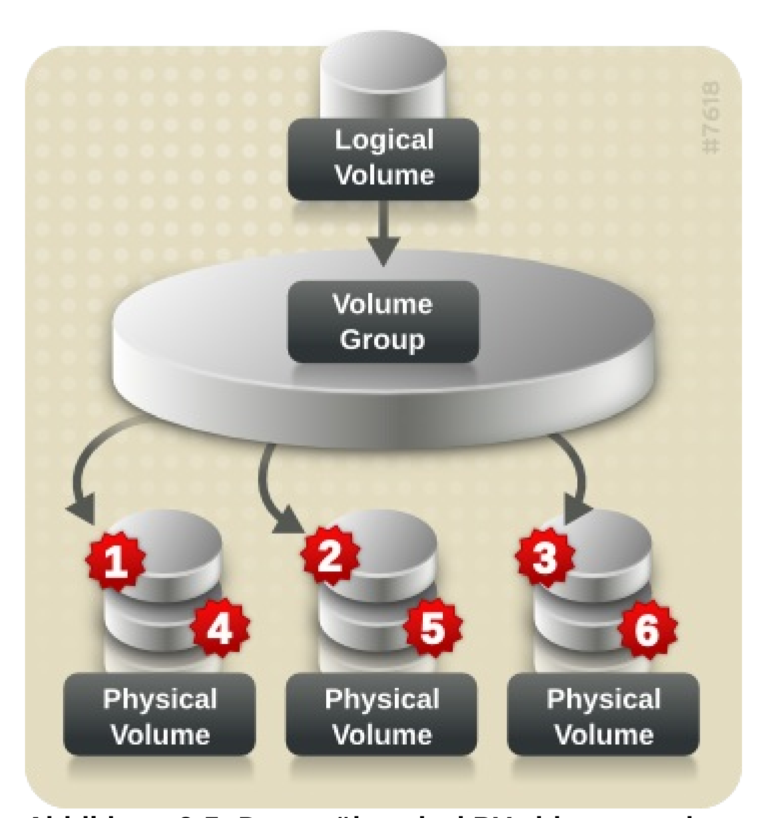



**Abbildung 2.5. Daten über drei PVs hinweg stripen**

Logische Striped-Datenträger können erweitert werden, indem ein weiteres Set an Geräten an das Ende des ersten Sets angehängt wird. Um jedoch einen logischen Striped-Datenträger erweitern zu können, muss auf den zugrunde liegenden physischen Datenträgern genügend freier Platz vorhanden sein, aus denen die Datenträgergruppe besteht, um den Stripe zu unterstützen. Wenn Sie beispielsweise einen wechselseitigen Stripe besitzen, der eine gesamte Datenträgergruppe beansprucht, ermöglicht Ihnen das Hinzufügen eines einzelnen physischen Datenträgers zu der Datenträgergruppe nicht das Erweitern des Stripes. Stattdessen müssen Sie mindestens zwei physische Datenträger zu der Datenträgergruppe hinzufügen. Werfen Sie einen Blick auf Abschnitt 4.4.12.1, „Striped-Datenträger vergrößern“ für weitere Informationen zur Erweiterung eines Striped-Datenträgers.

### 2.3.3. Gespiegelte logische Datenträger

Ein Mirror behält identische Kopien der Daten auf verschiedenen Geräten bei. Beim Schreiben von Daten auf ein Gerät werden diese gleichzeitig auf ein zweites Gerät geschrieben und so die Daten gespiegelt. Dies bietet einen gewissen Schutz vor Ausfällen von Geräten. Fällt ein Standbein des Mirrors aus, wird der logische Datenträger zu einem linearen Datenträger auf den weiterhin zugegriffen werden kann.

LVM unterstützt gespiegelte Datenträger. Beim Erstellen eines gespiegelten logischen Datenträgers stellt LVM sicher, dass Daten, die auf einen zugrunde liegenden physischen Datenträger geschrieben werden, auf einen separaten physischen Datenträger gespiegelt werden. Mithilfe von LVM können Sie gespiegelte logische Datenträger mit mehreren Mirrors erstellen.

Ein LVM-Mirror unterteilt das Gerät, das kopiert wird, in Bereiche, die typischerweise 512 KB groß sind. LVM pflegt eine kleine Protokolldatei, in der festgehalten wird, welche Bereiche mit dem (den) Mirror(s) synchron sind. Diese Protokolldatei kann auf der Platte gespeichert werden, so dass sie über Neustarts hinaus bestehen bleibt, oder aber im Speicher verbleiben.

Abbildung 2.6, „Gespiegelter logischer Datenträger“ zeigt einen gespiegelten logischen Datenträger mit einem Mirror. In dieser Konfiguration wird die Protokolldatei auf der Platte behalten.

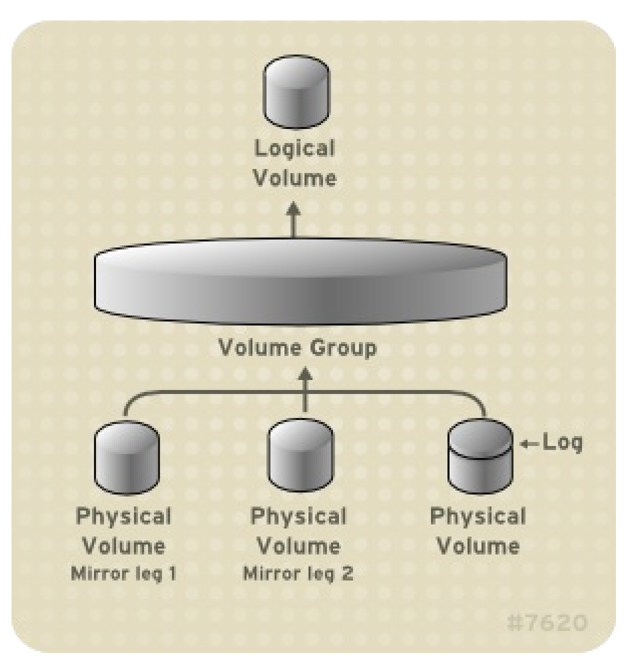



**Abbildung 2.6. Gespiegelter logischer Datenträger**

Werfen Sie einen Blick auf Abschnitt 4.4.3, „Gespiegelte Datenträger erstellen“ für Informationen zum Erstellen und Modifizieren von Mirrors.

### 2.3.4. Snapshot-Datenträger

Das LVM-Snapshot-Feature bietet die Fähigkeit, virtuelle Images eines Geräts zu einem bestimmten Moment zu erstellen, ohne eine Betriebsunterbrechung zu verursachen. Wenn eine Änderung an dem originalen Gerät (der Quelle) vorgenommen wird, nachdem ein Snapshot erstellt wurde, erstellt das Snapshot-Feature eine Kopie des geänderten Datenbereichs, wie vor der Änderung, so dass der Status des Geräts rekonstruiert werden kann.



**Anmerkung**

LVM-Snapshots werden zwischen den Knoten in einem Cluster nicht unterstützt. Sie können in einer gelusterten Datenträgergruppe keinen Snapshot-Datenträger erstellen.



**Anmerkung**

LVM-Snapshots werden für gespiegelte logische LVM-Datenträger nicht unterstützt.

Da Snapshot-Kopien nur die Datenbereiche kopieren, die sich nach der Erstellung des Snapshots ändern, erfordert das Snapshot-Feature lediglich eine sehr geringe Speichermenge. So sind beispielsweise bei einer selten aktualisierten Quelle 3-5 % der Kapazität der Quelle ausreichend, um den Snapshot beizubehalten.



**Anmerkung**

Snapshot-Kopien eines Dateisystems sind virtuelle Kopien und keine tatsächlichen Sicherungen von Medien für ein Dateisystem. Snapshots bieten keinen Ersatz für regelmäßige Backups.

Die Größe des Snapshots legt die Menge an Speicherplatz fest, die zum Speichern der Änderungen am originalen Datenträger vorgesehen ist. Wenn Sie beispielsweise einen Snapshot erzeugen und anschließend dessen Quelle vollständig überschreiben, müsste der Snapshot mindestens so groß wie der ursprüngliche Datenträger sein, um sämtliche Änderungen zu speichern. Sie sollten sich daher für den Umfang eines Snapshots danach richten, wie viele Änderungen Sie erwarten. So benötigt beispielsweise ein kurzlebiger Snapshot eines überwiegend gelesenen Datenträgers, wie z.B. **`/usr`**, weniger Speicherplatz als ein langlebiger Snapshot eines Datenträgers mit einer größeren Anzahl von Schreibvorgängen, wie z.B. **`/home`**.

Falls ein Snapshot vollläuft, wird er ungültig, da er die Änderungen am originalen Datenträger nicht länger nachverfolgen kann. Sie sollten die Größe des Snapshots regelmäßig überwachen. Snapshots können jedoch komplett in der Größe geändert werden, wenn Sie also die Speicherkapazität besitzen, können Sie die Größe des Snapshot-Datenträgers erhöhen, um zu verhindern, dass er verworfen wird. Umgekehrt können Sie die Größe des Datenträgers reduzieren, wenn Sie der Meinung sind, dass der Snapshot-Datenträger größer ist, als benötigt und so Platz freigeben, der von anderen logischen Datenträgern benötigt wird.

Beim Erstellen eines Snapshot-Dateisystems bleibt der komplette Lese- und Schreibzugriff auf die Quelle erhalten. Falls sich ein Teil des Snapshot verändert, wird dieser Teil markiert und niemals von dem ursprünglichen Datenträger kopiert.

Es gibt mehrere Verwendungszwecke für das Snapshot-Feature:

- Typischerweise wird ein Snapshot erstellt, wenn Sie eine Sicherung eines logischen Datenträgers durchführen müssen, ohne das laufende System anzuhalten, welches die Daten ständig aktualisiert.
- Sie können den Befehl **`fsck`** auf einem Snapshot-Dateisystem ausführen, um die Integrität des Dateisystems zu überprüfen und zu ermitteln, ob das originale Dateisystem eine Reparatur des Dateisystems erfordert.
- Da der Snapshot lesbar/schreibbar ist, können Sie Applikationen mit Produktionsdaten testen, indem Sie einen Snapshot erstellen und Tests für diesen Snapshot ausführen, wobei die tatsächlichen

Daten unberührt bleiben.

- Sie können LVM-Datenträger zur Verwendung mit Red Hat Virtualisierung erstellen. LVM-Snapshots können dazu verwendet werden, um Snapshots virtueller Gast-Images zu erstellen. Diese Snapshots bieten einen bequemen Weg zur Änderung vorhandener Gäste oder zur Erstellung neuer Gäste mit minimalem zusätzlichen Speicher. Weitere Informationen über das Erstellen von LVM-Snapshots virtueller Gäste finden Sie im *Red Hat Enterprise Linux Virtualisierungshandbuch*.

Werfen Sie einen Blick auf Abschnitt 4.4.4, „Snapshot-Datenträger erstellen“ für Informationen zum Erstellen von Snapshot-Datenträgern.

Ab der Red Hat Enterprise Linux 6 Release können Sie die **`--merge`**-Option des **`lvconvert`**-Befehls verwenden, um einen Snapshot wieder mit dem originalen Datenträger zusammenzuführen. Ein Anwendungsfall dieses Features ist ein Zurücksetzen des Systems, falls Sie Daten verloren haben oder aus anderen Gründen das System auf einen früheren Stand zurückversetzen müssen. Nach dem Zusammenführen mit dem Snapshot-Datenträger besitzt der daraus entstandene logische Datenträger den Namen, die Minor-Nummer und die UUID des originalen Datenträgers, und der Snapshot wird gelöscht. Weitere Informationen über die Verwendung dieser Option finden Sie in Abschnitt 4.4.5, „Snapshot-Datenträger zusammenführen“.

# Kapitel 3. Überblick über die LVM-Administration

Dieses Kapitel liefert einen Überblick über die administrativen Verfahren, mit denen Sie logische LVM-Datenträger konfigurieren können. Dieses Kapitel soll ein allgemeines Verständnis der nötigen Schritte vermitteln. Werfen Sie einen Blick auf Kapitel 5, *Konfigurationsbeispiele für LVM* für spezielle Schritt-für-Schritt-Beispiele von häufigen LVM-Konfigurationsverfahren.

Werfen Sie einen Blick auf Kapitel 4, *LVM-Administration mit CLI-Befehlen* für Beschreibungen von den CLI-Befehlen, die Sie zur LVM-Administration verwenden können. Alternativ können Sie auch das LVM-GUI verwenden, welches in Kapitel 7, *LVM-Administration mit dem LVM-GUI* näher beschrieben ist.

## 3.1. LVM-Datenträger in einem Cluster erstellen

Um logische Datenträger in einer Cluster-Umgebung zu erstellen, können Sie den Clustered Logical Volume Manager (CLVM) verwenden, die Erweiterungen zu LVM. Diese Erweiterungen erlauben es einem Cluster von Computern, mithilfe von LVM gemeinsam verwendeten Speicher (z.B. auf einem SAN) zu verwalten. Um CLVM zu verwenden, muss die Hochverfügbarkeits-Add-On und Resilient Storage Add-On Software einschließlich des clvmd-Daemons beim Booten gestartet werden, wie in Abschnitt 1.4, „Der Clustered Logical Volume Manager (CLVM)“ erläutert.

Das Erstellen von logischen LVM-Datenträgern in einer Cluster-Umgebung ist identisch zu der Erstellung von logischen LVM-Datenträgern auf einem einzelnen Knoten. Es gibt keine Unterschiede bei den LVM-Befehlen selbst oder bei der grafischen Benutzeroberfläche von LVM. Um die LVM-Datenträger, die Sie in einem Cluster erstellen zu aktivieren, muss die Cluster-Infrastruktur ausgeführt werden und einsatzfähig sein.

CLVM erfordert Änderungen an der **lvm.conf**-Datei zur clusterweiten Sperrung. Sie finden Informationen über die Konfiguration der **lvm.conf**-Datei zur Unterstützung von Cluster-Sperrung in der **lvm.conf**-Datei selbst. Informationen über die**lvm.conf**-Datei finden Sie in Anhang B, *Die LVM-Konfigurationsdateien*.

Standardmäßig sind logische Datenträger, die mit CLVM auf gemeinsam verwendetem Speicher erstellt wurden, für alle Systeme sichtbar, die Zugriff auf diesen gemeinsam verwendeten Speicher haben. Es ist möglich, Datenträgergruppen zu erstellen, in denen alle enthaltene Speichergeräte nur für einen Knoten im Cluster sichtbar sind. Es ist ebenfalls möglich, den Status einer Datenträgergruppe von einer lokalen Datenträgergruppe zu einer geclusterten Datenträgergruppe zu ändern. Weitere Informationen diesbezüglich finden Sie in Abschnitt 4.3.2, „Datenträgergruppen in einem Cluster erstellen“ and Abschnitt 4.3.7, „Parameter einer Datenträgergruppe verändern“.



**Warnung**

Wenn Sie mit CLVM Datenträgergruppen auf gemeinsam verwendetem Speicher erzeugen, stellen Sie sicher, dass alle Knoten im Cluster Zugriff auf die physischen Datenträger haben, aus denen sich die Datenträgergruppe zusammensetzt. Asymmetrische Cluster-Konfigurationen, bei denen einige Knoten Zugriff auf den Speicher haben und andere nicht, werden nicht unterstützt.

Werfen Sie einen Blick auf *Cluster-Administration* für Informationen zur Installation des Hochverfügbarkeits-Add-Ons und zur Einrichtung der Cluster-Infrastruktur.

Ein Beispiel zur Erstellung eines gespiegelten logischen Datenträgers in einem Cluster finden Sie in Abschnitt 5.5, „Erstellen eines gespiegelten logischen LVM-Datenträgers in einem Cluster“.

## 3.2. Überblick über die Erstellung eines logischen Datenträgers

Nachfolgend ist eine Zusammenfassung der Schritte aufgeführt, die zur Erstellung eines logischen LVM-Datenträgers nötig sind.

1. Initialisieren Sie die Partitionen, die Sie als physische Datenträger für den LVM-Datenträger verwenden werden (dies kennzeichnet diese).
2. Erstellen Sie einen Datenträgergruppe.
3. Erstellen Sie einen logischen Datenträger.

Nach der Erstellung des logischen Datenträgers können Sie das Dateisystem erstellen und einhängen. Die Beispiele in diesem Dokument verwenden GFS2-Dateisysteme.



**Anmerkung**

Obwohl ein GFS2-Dateisystem in einem eigenständigen System oder als Teil einer Cluster-Konfiguration implementiert werden kann, unterstützt Red Hat für die Red Hat Enterprise 6 Release nicht den Einsatz von GFS2 als Ein-Knoten-Dateisystem. Red Hat unterstützt jedoch nach wie vor Ein-Knoten-GFS2-Dateisysteme zum Einhängen von Snapshots von Cluster-Dateisystemen (z.B. zwecks Backup).

1. Erstellen Sie mit dem Befehl **`mkfs.gfs2`** ein GFS2-Dateisystem auf dem logischen Datenträger.
2. Erstellen Sie mit dem Befehl **`mkdir`** einen neuen Einhängepunkt. Erstellen Sie in einem Cluster-System die Einhängepunkte auf allen Knoten im Cluster.
3. Hängen Sie das Dateisystem ein. Sie sollten ggf. für jedes System eine Zeile zur **`fstab`**-Datei hinzufügen.

Alternativ können Sie das GFS2-Dateisystem mit der grafischen Benutzeroberfläche des LVM erstellen und einhängen.

Das Erstellen des LVM-Datenträgers ist unabhängig vom Rechner, da sich der Speicherbereich für die Informationen der LVM-Einrichtung auf den physischen Datenträgern und nicht auf dem Rechner, auf der der Datenträger erstellt wurde, befindet. Server, die den Speicher verwenden, haben lokale Kopien, können diese jedoch auch vom Inhalt der physischen Datenträger neu erstellen. Falls die LVM-Versionen kompatibel sind, können Sie physische Datenträger mit einem anderen Server verknüpfen.

## 3.3. Vergrößern eines Dateisystems auf einem logischen Datenträger

Führen Sie die folgende Schritte durch, um ein Dateisystem auf einem logischen Datenträger zu vergrößern:

1. Erstellen Sie einen neuen physischen Datenträger.
2. Erweitern Sie die Datenträgergruppe, die den logischen Datenträger enthält mit dem Dateisystem, das Sie vergrößern, um den neuen physischen Datenträger einzubinden.
3. Erweitern Sie den logischen Datenträger, der den neuen physischen Datenträger einbinden soll.
4. Vergrößern Sie das Dateisystem.

Falls Sie über genügend nicht zugewiesenen Platz in der Datenträgergruppe verfügen, können Sie diesen Platz zur Erweiterung des logischen Datenträgers verwenden, anstatt die Schritte 1 und 2 durchzuführen.

## 3.4. Backup eines logischen Datenträgers

Backups von Metadaten und Archiven werden automatisch bei jeder Änderung der Konfiguration von Datenträgergruppen und logischen Datenträgern erstellt, sofern dies nicht in der Datei **lvm.conf** deaktiviert wird. Standardmäßig wird die Sicherung der Metadaten in **/etc/lvm/backup** und die Sicherung der Metadaten-Archive in **/etc/lvm/archive** gespeichert. Die Dauer der Speicherung von Metadaten-Archiven in **/etc/lvm/archive** und die Anzahl der Dateien, die gespeichert werden sollen, werden durch Parameter festgelegt, die Sie in der Datei **lvm.conf** setzen können. Ein tägliches Backup des Systems sollte den Inhalt des Verzeichnisses **/etc/lvm** umfassen.

Beachten Sie, dass eine Sicherung der Metadaten nicht die Benutzer- und Systemdaten sichert, die sich auf den logischen Datenträgern befinden.

Sie können die Metadaten manuell mit dem Befehl **vgcfgbackup** in der Datei **/etc/lvm/backup** sichern. Mithilfe des Befehls **vgcfgrestore** können Sie die Metadaten wiederherstellen. Die Befehle **vgcfgbackup** und **vgcfgrestore** werden in Abschnitt 4.3.12, „Metadaten von Datenträgergruppen sichern“ beschrieben.

## 3.5. Protokollierung

Alle Meldungen, die ausgegeben werden, passieren ein Protokollierungsmodul mit unabhängiger Auswahl der Protokollierungsebene für:

- Standard-Ausgabefehler
- syslog
- Protokolldatei
- externe Protokollierung

Die Protokollierungsebenen werden in der Datei **/etc/lvm/lvm.conf** gesetzt, wie in Anhang B, *Die LVM-Konfigurationsdateien* beschrieben.

# Kapitel 4. LVM-Administration mit CLI-Befehlen

Dieses Kapitel fasst die einzelnen administrativen Aufgaben zusammen, die Sie mit den Befehlen des LVM Command Line Interface (CLI, die Befehlszeile) durchführen können, um logische Datenträger zu erstellen und zu pflegen.



**Anmerkung**

Wenn Sie einen LVM-Datenträger für eine geclusterte Umgebung erstellen oder verändern, müssen Sie sicherstellen, dass der **clvmd**-Daemon läuft. Weitere Informationen diesbezüglich finden Sie in Abschnitt 3.1, „LVM-Datenträger in einem Cluster erstellen“.

## 4.1. Verwendung von CLI-Befehlen

Einige Eigenschaften sind allen LVM-CLI-Befehlen gemein.

Wenn Größen bei der Angabe auf der Befehlszeile benötigt werden, können Einheiten immer explizit angegeben werden. Falls Sie keine Einheit angeben, wird ein Standardwert angenommen, normalerweise KB oder MB. LVM-CLI-Befehle akzeptieren keine Brüche.

Bei der Angabe von Einheiten im Rahmen eines Befehlszeilenparameters achtet LVM nicht auf Groß- und Kleinschreibung. Die Angabe von "M" oder "m" beispielsweise ist identisch und es werden Zweierpotenzen (Vielfaches von 1.024) verwendet. Wird jedoch der Parameter **--units** in einem Befehl angegeben, weist die Kleinschreibung darauf hin, dass Einheiten ein Vielfaches von 1.024 sind, während die Großschreibung verdeutlicht, dass es sich um Einheiten mit einem Vielfachen von 1.000 handelt.

Wenn Befehle Datenträgergruppen oder logische Datenträgernamen als Parameter verwenden, ist die Angabe des vollständigen Pfades optional. Ein logischer Datenträger mit der Bezeichnung **lvol0** in einer Datenträgergruppe mit der Bezeichnung **vg0** kann als **vg0/lvol0** angegeben werden. Wo eine Liste von Datenträgergruppen erforderlich ist, jedoch freigelassen wird, wird stattdessen eine Liste aller Datenträgergruppen an die Stelle gesetzt. Wo eine Liste von logischen Datenträgern erforderlich ist, jedoch eine Datenträgergruppe angegeben wird, wird stattdessen eine Liste aller logischen Datenträger in dieser Datenträgergruppe an die Stelle gesetzt. So zeigt beispielsweise der Befehl **lvdisplay vg0** alle logischen Datenträger in der Datenträgergruppe **vg0** an.

Alle LVM-Befehle akzeptieren einen **-v**-Parameter, der mehrfach eingegeben werden kann, um die Ausführlichkeit der Ausgabe zu erhöhen. Die folgenden Beispiele zeigen beispielsweise die standardmäßige Ausgabe des Befehls **lvcreate**.

```
# lvcreate -L 50MB new_vg
  Rounding up size to full physical extent 52.00 MB
  Logical volume "lvol0" created
```

Der folgende Befehl zeigt die Ausgabe des Befehls **lvcreate** mit dem Parameter **-v**.

```
# lvcreate -v -L 50MB new_vg
    Finding volume group "new_vg"
  Rounding up size to full physical extent 52.00 MB
    Archiving volume group "new_vg" metadata (seqno 4).
    Creating logical volume lvol0
    Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 5).
    Found volume group "new_vg"
    Creating new_vg-lvol0
    Loading new_vg-lvol0 table
    Resuming new_vg-lvol0 (253:2)
    Clearing start of logical volume "lvol0"
    Creating volume group backup "/etc/lvm/backup/new_vg" (seqno 5).
  Logical volume "lvol0" created
```

Mithilfe der Parameter **-vv**, **-vvv** oder **-vvvv** können Sie außerdem umfangreichere Details zur Ausführung des Befehls anzeigen. Der Parameter **-vvvv** liefert an dieser Stelle die meisten Informationen. Das folgende Beispiel zeigt lediglich die ersten paar Zeilen der Ausgabe für den Befehl **lvcreate** mit dem Parameter **-vvvv**.

```
# lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:913          Processing: lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:916          O_DIRECT will be used
#config/config.c:864        Setting global/locking_type to 1
#locking/locking.c:138        File-based locking selected.
#config/config.c:841        Setting global/locking_dir to /var/lock/lvm
#activate/activate.c:358        Getting target version for linear
#ioctl/libdm-iface.c:1569          dm version   OF   [16384]
#ioctl/libdm-iface.c:1569          dm versions   OF   [16384]
#activate/activate.c:358        Getting target version for striped
#ioctl/libdm-iface.c:1569          dm versions   OF   [16384]
#config/config.c:864        Setting activation/mirror_region_size to 512
...
```

Mithilfe des Parameters **--help** eines Befehls können Sie sich den Hilfebildschirm von jedem LVM-CLI-Befehl anzeigen lassen.

```
Befehlsname --help
```

Führen Sie den Befehl **man** aus, um die Handbuchseite für einen Befehl anzuzeigen:

```
man Befehlsname
```

Der Befehl **man lvm** liefert allgemeine Online-Informationen über LVM.

Alle LVM-Objekte werden intern mit einer UUID referenziert, die bei der Erstellung des Objekts zugewiesen wird. Dies kann in einer Situation nützlich sein, in der Sie einen physischen Datenträger mit der Bezeichnung **/dev/sdf**, der Teil einer Datenträgergruppe ist, entfernen und feststellen, dass dieser nun unter **/dev/sdk** erscheint, wenn Sie ihn wieder einhängen. LVM findet den physischen Datenträger auch weiterhin, da es diesen anhand dessen UUID und nicht dessen Gerätenamen identifiziert. Werfen Sie einen Blick auf Abschnitt 6.4, „Wiederherstellen von Metadaten eines physischen Datenträgers“ für Informationen zur Angabe der UUID eines physischen Datenträgers bei dessen Erstellung.

## 4.2. Administration von physischen Datenträgern

Dieser Abschnitt beschreibt die Befehle, mit denen Sie die verschiedenen Aspekte der Administration von physischen Datenträgern durchführen.

## 4.2.1. Physische Datenträger erstellen

Die folgenden Unterabschnitte beschreiben die Befehle, die zur Erstellung von physischen Datenträgern verwendet werden.

### 4.2.1.1. Partitionstyp einstellen

Falls Sie ein ganzes Festplattengerät für Ihren physischen Datenträger verwenden, darf die Platte keine Partitionstabelle enthalten. Für DOS-Festplattenpartitionen sollte die Partitions-ID mithilfe der Befehle **fdisk** oder **cfdisk** (oder entsprechenden Befehlen) auf 0x8e gesetzt werden. Für ganze Festplattengeräte muss lediglich die Partitionstabelle gelöscht werden, was alle Daten auf dieser Platte nachhaltig zerstört. Sie können eine bestehende Partitionstabelle entfernen, indem Sie den ersten Sektor mit dem folgenden Befehl mit Nullen überschreiben:

```
dd if=/dev/zero of=Physischer Datenträger bs=512 count=1
```

### 4.2.1.2. Physische Datenträger initialisieren

Verwenden Sie den Befehl **pvcreate**, um ein Blockgerät zu initialisieren, das als physischer Datenträger verwendet werden soll. Die Initialisierung entspricht einer Formatierung eines Dateisystems.

Der folgende Befehl initialisiert **/dev/sdd1**, **/dev/sde1** und **/dev/sdf1**, so dass diese als physische LVM-Datenträger verwendet werden können.

```
pvcreate /dev/sdd1 /dev/sde1 /dev/sdf1
```

Um statt der gesamten Platte einzelne Partitionen zu initialisieren, führen Sie **pvcreate** auf der Partition aus. Das folgende Beispiel initialisiert **/dev/hdb1** als einen physischen LVM-Datenträger für die spätere Verwendung als Teil eines logischen LVM-Datenträgers.

```
pvcreate /dev/hdb1
```

### 4.2.1.3. Suche nach Blockgeräten

Mit dem Befehl **lvmdiskscan** können Sie nach Blockgeräten suchen, die ggf. als physische Datenträger genutzt werden können, wie im folgenden Beispiel gezeigt.

```
# lvmdiskscan
/dev/ram0                    [       16.00 MB]
  /dev/sda                   [       17.15 GB]
  /dev/root                  [       13.69 GB]
  /dev/ram                   [       16.00 MB]
  /dev/sda1                  [       17.14 GB] LVM physical volume
  /dev/VolGroup00/LogVol01   [      512.00 MB]
  /dev/ram2                  [       16.00 MB]
  /dev/new_vg/lvol0          [       52.00 MB]
  /dev/ram3                  [       16.00 MB]
  /dev/pkl_new_vg/sparkie_lv [        7.14 GB]
  /dev/ram4                  [       16.00 MB]
  /dev/ram5                  [       16.00 MB]
  /dev/ram6                  [       16.00 MB]
  /dev/ram7                  [       16.00 MB]
  /dev/ram8                  [       16.00 MB]
  /dev/ram9                  [       16.00 MB]
  /dev/ram10                 [       16.00 MB]
  /dev/ram11                 [       16.00 MB]
  /dev/ram12                 [       16.00 MB]
  /dev/ram13                 [       16.00 MB]
  /dev/ram14                 [       16.00 MB]
  /dev/ram15                 [       16.00 MB]
  /dev/sdb                   [       17.15 GB]
  /dev/sdb1                  [       17.14 GB] LVM physical volume
  /dev/sdc                   [       17.15 GB]
  /dev/sdc1                  [       17.14 GB] LVM physical volume
  /dev/sdd                   [       17.15 GB]
  /dev/sdd1                  [       17.14 GB] LVM physical volume
  7 disks
  17 partitions
  0 LVM physical volume whole disks
  4 LVM physical volumes
```

### 4.2.2. Physische Datenträger anzeigen

Es gibt drei Befehle, die Sie für das Anzeigen der Eigenschaften von physischen LVM-Datenträgern verwenden können: **pvs**, **pvdisplay** und **pvscan**.

Der Befehl **pvs** liefert Informationen zu einem physischen Datenträger in einer konfigurierbaren Form, wobei eine Zeile pro physischem Datenträger angezeigt wird. Der **pvs**-Befehl bietet ein hohes Maß an Formatkontrolle und ist nützlich für das Skripting. Werfen Sie einen Blick auf Abschnitt 4.8, „Angepasste Berichterstattung für LVM“ für Informationen zur Verwendung des Befehls **pvs** zur Anpassung Ihrer Ausgabe.

Der Befehl **pvdisplay** bietet eine umfangreiche, mehrzeilige Ausgabe für jeden physischen Datenträger. Er zeigt physische Eigenschaften (Größe, Extents, Datenträgergruppe, etc.) in einem festen Format an.

Das folgende Beispiel zeigt die Ausgabe des Befehls **pvdisplay** für einen einzelnen physischen Datenträger.

```
# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sdc1
  VG Name               new_vg
  PV Size               17.14 GB / not usable 3.40 MB
  Allocatable           yes
  PE Size (KByte)       4096
  Total PE              4388
  Free PE               4375
  Allocated PE          13
  PV UUID               Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
```

Der Befehl **pvscan** sucht alle unterstützten LVM-Blockgeräte im System nach physischen Datenträgern ab.

Der folgende Befehl zeigt alle gefundenen physischen Geräte an:

```
# pvscan
 PV /dev/sdb2   VG vg0   lvm2 [964.00 MB / 0   free]
 PV /dev/sdc1   VG vg0   lvm2 [964.00 MB / 428.00 MB free]
 PV /dev/sdc2            lvm2 [964.84 MB]
 Total: 3 [2.83 GB] / in use: 2 [1.88 GB] / in no VG: 1 [964.84 MB]
```

In der Datei **lvm.conf** können Sie einen Filter definieren, damit dieser Befehl bestimmte physische Datenträger nicht absucht. Werfen Sie einen Blick auf Abschnitt 4.5, „LVM-Geräte-Scans mit Filtern kontrollieren“ für Informationen zur Verwendung von Filtern zur Steuerung, welche Geräte abgesucht werden.

### 4.2.3. Zuweisung auf einem physischen Datenträger verhindern

Mithilfe des Befehls **pvchange** können Sie die Zuweisung von Extents auf freien Speicherplatz von einem oder mehreren physischen Datenträgern verhindern. Dies ist ggf. bei Festplattenfehlern oder beim Entfernen des physischen Datenträgers notwendig.

Der folgende Befehl unterbindet die Zuweisung von physischen Extents auf **/dev/sdk1**.

```
pvchange -x n /dev/sdk1
```

Sie können die Parameter **-xy** des Befehls **pvchange** verwenden, um eine Zuweisung zu gestatten, wenn diese zuvor verweigert wurde.

### 4.2.4. Größe eines physischen Datenträgers anpassen

Falls Sie aus irgendeinem Grund die Größe eines zugrunde liegenden Blockgeräts anpassen müssen, verwenden Sie den Befehl **pvresize**, um LVM mit der neuen Größe zu aktualisieren. Sie können diesen Befehl ausführen, während LVM den physischen Datenträger verwendet.

### 4.2.5. Physische Datenträger entfernen

Falls ein Gerät nicht länger für die Verwendung mit LVM benötigt wird, können Sie das LVM-Label mit dem Befehl **pvremove** entfernen. Durch die Ausführung von **pvremove** werden die LVM-Metadaten auf einem leeren physischen Datenträger auf Null gesetzt.

Falls der physische Datenträger, den Sie entfernen möchten, derzeit Teil einer Datenträgergruppe ist, müssen Sie ihn mit dem Befehl **vgreduce** aus der Datenträgergruppe entfernen, wie in Abschnitt 4.3.6, „Physische Datenträger aus einer Datenträgergruppe entfernen“ beschrieben.

```
# pvremove /dev/ram15
  Labels on physical volume "/dev/ram15" successfully wiped
```

## 4.3. Administration von Datenträgergruppen

Dieser Abschnitt beschreibt die Befehle, die die verschiedenen Aspekte der Administration von Datenträgergruppen verrichten.

### 4.3.1. Datenträgergruppen erstellen

Verwenden Sie den Befehl **vgcreate** zur Erstellung einer Datenträgergruppe aus einem oder mehreren physischen Datenträgern. Der Befehl **vgcreate** erstellt eine neue Datenträgergruppe mit neuem Namen und fügt dieser mindestens einen physischen Datenträger hinzu.

Der folgende Befehl erstellt eine Datenträgergruppe mit dem Namen **vg1**, welche die physischen Datenträger **/dev/sdd1** und **/dev/sde1** beinhaltet.

```
vgcreate vg1 /dev/sdd1 /dev/sde1
```

Werden physische Datenträger für das Erstellen einer Datenträgergruppe verwendet, wird der Plattenplatz standardmäßig in 4 MB große Extents aufgeteilt. Diese Extents repräsentieren die minimale Größe, um die ein logischer Datenträger vergrößert, bzw. verkleinert werden kann. Eine große Anzahl an Extents hat keine Auswirkungen auf I/O-Leistung des logischen Datenträgers.

Mithilfe der **-s**-Option des **vgcreate**-Befehls können Sie die Extent-Größe angeben, falls der Standardwert nicht passt. Sie können die Anzahl der physischen oder logischen Datenträger, die die Datenträgergruppe umfassen kann, einschränken, indem Sie die Parameter **-p** und **-l** des **vgcreate**-Befehls verwenden.

Standardmäßig weist eine Datenträgergruppe physische Extents nach selbstverständlichen Regeln zu, z.B. keine parallelen Stripes auf demselben physischen Datenträger. Dies ist die **normal** Zuweisungsrichtlinie. Sie können den Parameter **--alloc** des **vgcreate**-Befehls verwenden, um die Zuweisungsrichtlinien auf **contiguous**, **anywhere** oder **cling** zu setzen.

Die **contiguous**-Richtlinie erfordert, dass neue Extents neben existierenden Extents liegen. Falls genügend freie Extents vorhanden sind, um einer Zuweisungsanfrage zu entsprechen, die **normal**-Zuweisungsrichtlinie diese jedoch nicht verwenden würde, beansprucht die **anywhere**-Zuweisungsrichtlinie diese, indem sie zwei Stripes auf demselben physischen Datenträger platziert. Die **cling**-Richtlinie platziert neue Extents auf demselben physischen Datenträger, wie vorhandene Extents in demselben Stripe auf dem logischen Datenträger. Diese Richtlinien können mithilfe des Befehls **vgchange** geändert werden.

Unter Abschnitt 4.4.12.2, „Erweitern eines logischen Datenträgers mit der **cling**-Zuweisungsrichtlinie“ finden Sie Informationen über die Verwendung der **cling**-Richtlinie in Verbindung mit LVM-Tags, um beim Erweitern eines LVM-Datenträgers die zu verwendenden zusätzlichen physischen Datenträger zu spezifizieren.

Im Allgemeinen werden andere Zuweisungsrichtlinien als **normal** nur in speziellen Fällen benötigt, in denen Sie unübliche oder nicht standardmäßige Extent-Zuweisungen bestimmen müssen.

LVM-Datenträgergruppen und zugrunde liegende logische Datenträger sind im Verzeichnisbaum für spezielle Gerätedateien im Verzeichnis **/dev** mit dem folgenden Aufbau eingebunden:

```
/dev/vg/lv/
```

Wenn Sie beispielsweise zwei Datenträgergruppen **myvg1** und **myvg2** erstellen und jede mit drei logischen Datenträgern mit der Bezeichnung **lvo1**, **lvo2** und **lvo3**, erstellt dies sechs spezielle Gerätedateien:

```
/dev/myvg1/lv01
/dev/myvg1/lv02
/dev/myvg1/lv03
/dev/myvg2/lv01
/dev/myvg2/lv02
/dev/myvg2/lv03
```

Die maximale Gerätegröße bei LVM beträgt 8 Exabytes auf 64-Bit CPUs.

### 4.3.2. Datenträgergruppen in einem Cluster erstellen

Sie können Datenträgergruppen in einer Cluster-Umgebung mithilfe des Befehls **vgcreate** erstellen, genau wie Sie diese auch auf einem einzelnen Knoten erstellen.

Standardmäßig sind logische Datenträger, die mit CLVM auf gemeinsam verwendetem Speicher erstellt wurden, für alle Systeme sichtbar, die Zugriff auf diesen gemeinsam verwendeten Speicher haben. Es ist jedoch möglich, mithilfe der **-c n**-Option des **vgcreate**-Befehls Datenträgergruppen zu erstellen, die nur für einen Knoten im Cluster sichtbar sind.

Wird der folgende Befehl in einer Cluster-Umgebung ausgeführt, erstellt er eine Datenträgergruppe, die lokal ist für den Knoten, auf dem der Befehl ausgeführt wurde. Der Befehl erstellt einen lokalen Datenträger namens **vg1**, der die physischen Datenträger **/dev/sdd1** und **/dev/sde1** enthält.

```
vgcreate -c n vg1 /dev/sdd1 /dev/sde1
```

Mithilfe der **-c**-Option des **vgchange**-Befehls können Sie ändern, ob eine vorhandene Datenträgergruppe lokal oder geclustert ist, wie in Abschnitt 4.3.7, „Parameter einer Datenträgergruppe verändern“ beschrieben.

Mithilfe des **vgs**-Befehls können Sie überprüfen, ob eine vorhandene Datenträgergruppe geclustert ist; ist der Datenträger geclustert, wird der **c**-Parameter angezeigt. Der folgende Befehl zeigt die Parameter der Datenträgergruppen **VolGroup00** und **testvg1** an. In diesem Beispiel ist **VolGroup00** nicht geclustert, wohingegen **testvg1** geclustert ist, wie der **c**-Parameter unter der **Attr**-Überschrift anzeigt.

```
[root@doc-07]# vgs
  VG            #PV #LV #SN Attr   VSize  VFree
  VolGroup00      1   2   0 wz--n- 19.88G    0
  testvg1         1   1   0 wz--nc 46.00G 8.00M
```

Weitere Informationen über den **vgs**-Befehl finden Sie in Abschnitt 4.3.4, „Datenträgergruppen anzeigen“, Abschnitt 4.8, „Angepasste Berichterstattung für LVM“ sowie auf der **vgs**-Handbuchseite.

### 4.3.3. Physische Datenträger zu einer Datenträgergruppe hinzufügen

Um zusätzliche physische Datenträger zu einer bestehenden Datenträgergruppe hinzuzufügen, verwenden Sie den Befehl **vgextend**. Der Befehl **vgextend** erhöht die Kapazität einer Datenträgergruppe, indem eine oder mehrere freie physische Datenträger hinzugefügt werden.

Der folgende Befehl fügt den physischen Datenträger **/dev/sdf1** zur Datenträgergruppe **vg1** hinzu.

```
vgextend vg1 /dev/sdf1
```

### 4.3.4. Datenträgergruppen anzeigen

Es gibt zwei Befehle, mit denen Sie die Eigenschaften von LVM-Datenträgergruppen anzeigen können: **vgs** und **vgdisplay**.

Der Befehl **vgscan** zeigt auch die Datenträgergruppen an, obwohl sein primärer Zweck das Absuchen aller Platten nach Datenträgergruppen und das Neuerstellen der LVM-Cache-Datei ist. Werfen Sie einen Blick auf Abschnitt 4.3.5, „Platten nach Datenträgergruppen zum Erstellen der Cache-Datei absuchen“ für weitere Informationen zum Befehl **vgscan**.

Der Befehl **vgs** liefert Informationen zu Datenträgergruppen in einer konfigurierbaren Form, wobei eine Zeile pro Datenträgergruppe angezeigt wird. Der **vgs**-Befehl bietet ein hohes Maß an Formatkontrolle und ist nützlich für das Skripting. Werfen Sie einen Blick auf Abschnitt 4.8, „Angepasste Berichterstattung für LVM“ für Informationen zur Verwendung des Befehls **vgs** zur Anpassung Ihrer Ausgabe.

Der Befehl **vgdisplay** zeigt die Eigenschaften einer Datenträgergruppe (wie Größe, Extents, Anzahl der physischen Datenträger, etc.) in einem festen Format an. Das folgende Beispiel zeigt die Ausgabe des **vgdisplay**-Befehls für die Datenträgergruppe **new_vg**. Wenn Sie keine Datenträgergruppe angeben, werden alle vorhandenen Gruppen angezeigt.

```
# vgdisplay new_vg
  --- Volume group ---
  VG Name               new_vg
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  11
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               51.42 GB
  PE Size               4.00 MB
  Total PE              13164
  Alloc PE / Size       13 / 52.00 MB
  Free  PE / Size       13151 / 51.37 GB
  VG UUID               jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

### 4.3.5. Platten nach Datenträgergruppen zum Erstellen der Cache-Datei absuchen

Der Befehl **vgscan** sucht alle unterstützten Plattengeräte im System nach physischen LVM-Datenträgern und Datenträgergruppen ab. Dies erstellt die LVM-Cache-Datei in **/etc/lvm/.cache**, welche eine Liste von aktuellen LVM-Geräten unterhält.

LVM führt den **vgscan**-Befehl beim Systemstart und zu anderen Zeitpunkten während des LVM-Betriebs automatisch aus, z.B. wenn Sie den **vgcreate**-Befehl ausführen oder wenn LVM eine Inkonsistenz entdeckt.



**Anmerkung**

Bei einer Veränderung Ihrer Hardware-Konfiguration müssen Sie **vgscan** ggf. manuell ausführen, was zur Folge hat, dass neue Geräte für das System sichtbar werden, die zum Zeitpunkt des Systemstarts noch nicht existierten. Dies kann beispielsweise ggf. dann notwendig werden, wenn Sie in einem SAN neue Platten zum System hinzufügen oder eine neue Platte, die als physischer Datenträger gekennzeichnet war, im laufenden Betrieb austauschen.

In der Datei **lvm.conf** können Sie einen Filter definieren, damit dieser Befehl spezielle Geräte nicht absucht. Werfen Sie einen Blick auf Abschnitt 4.5, „LVM-Geräte-Scans mit Filtern kontrollieren“ für Informationen zur Verwendung von Filtern zur Kontrolle, welche Geräte abgesucht werden.

Das folgende Beispiel zeigt die Ausgabe eines **vgscan**-Befehls.

```
# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "new_vg" using metadata type lvm2
  Found volume group "officevg" using metadata type lvm2
```

### 4.3.6. Physische Datenträger aus einer Datenträgergruppe entfernen

Verwenden Sie den Befehl **vgreduce**, um nicht benutzte physische Datenträger aus einer Datenträgergruppe zu entfernen. Der Befehl **vgreduce** verkleinert die Kapazität einer Datenträgergruppe durch Entfernen von einem oder mehreren leeren physischen Datenträgern. Dies setzt die physischen Datenträger frei, die in unterschiedlichen Datenträgergruppen verwendet werden, oder vom System gelöscht werden sollen.

Vor dem Entfernen eines physischen Datenträgers aus einer Datenträgergruppe können Sie mithilfe des Befehls **pvdisplay** sicherstellen, dass der physische Datenträger von keinem logischen Datenträger verwendet wird.

```
# pvdisplay /dev/hda1

-- Physical volume ---
PV Name               /dev/hda1
VG Name               myvg
PV Size               1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV#                   1
PV Status            available
Allocatable           yes (but full)
Cur LV                1
PE Size (KByte)       4096
Total PE              499
Free PE               0
Allocated PE          499
PV UUID               Sd44tK-9IRw-SrMC-MOkn-76iP-iftz-OVSen7
```

Falls der physische Datenträger noch verwendet wird, müssen Sie die Daten mithilfe des Befehls **pvmove** auf einen anderen physischen Datenträger migrieren. Verwenden Sie anschließend den Befehl **vgreduce**, um den physischen Datenträger zu entfernen.

Der folgende Befehl entfernt den physischen Datenträger **/dev/hda1** aus der Datenträgergruppe **my_volume_group**.

```
# vgreduce my_volume_group /dev/hda1
```

### 4.3.7. Parameter einer Datenträgergruppe verändern

Der **vgchange**-Befehl wird dazu verwendet, um Datenträgergruppen zu aktivieren bzw. zu deaktivieren, wie in Abschnitt 4.3.8, „Datenträgergruppen aktivieren und deaktivieren“ beschrieben. Sie können diesen Befehl auch dazu verwenden, um mehrere Parameter für eine vorhandene Datenträgergruppe zu verändern.

Der folgende Befehl ändert die maximale Anzahl an logischen Datenträgern der Datenträgergruppe **vg00** auf 128.

```
vgchange -l 128 /dev/vg00
```

Werfen Sie einen Blick auf die Handbuchseite von **vgchange**(8) für eine Beschreibung der Parameter der Datenträgergruppe, die Sie mit dem Befehl **vgchange** ändern können.

### 4.3.8. Datenträgergruppen aktivieren und deaktivieren

Beim Erstellen einer Datenträgergruppe ist diese standardmäßig aktiviert. Dies bedeutet, dass die logischen Datenträger in dieser Gruppe gelesen und ggf. verändert werden können.

Es gibt verschiedene Umstände, unter denen Sie eine Datenträgergruppe auf inaktiv setzen müssen, um sie für den Kernel nicht erkenntlich zu machen. Verwenden Sie den Parameter **-a** (**--available**) des Befehls **vgchange**, um eine Datenträgergruppe zu aktivieren, bzw. zu deaktivieren.

Das folgende Beispiel deaktiviert die Datenträgergruppe **my_volume_group**.

```
vgchange -a n my_volume_group
```

Falls Cluster-Sperrung aktiviert ist, fügen Sie ’e’ hinzu, um eine Datenträgergruppe exklusiv auf einem Knoten zu aktivieren oder zu deaktivieren oder ’l’, um eine Datenträgergruppe nur auf dem lokalen Knoten zu aktivieren, bzw. zu deaktivieren. Logische Datenträger mit Single-Host-Snapshots werden immer exklusiv aktiviert, weil sie nur auf einem Knoten auf einmal verwendet werden können.

Mit dem Befehl **lvchange** können Sie einzelne logische Datenträger wie unter Abschnitt 4.4.8, „Parameter einer logischen Datenträgergruppe ändern“ beschrieben, deaktivieren. Werfen Sie einen Blick auf Abschnitt 4.7, „Logische Datenträger auf einzelnen Knoten in einem Cluster aktivieren“ für Informationen zur Aktivierung von logischen Datenträgern auf einzelnen Knoten in einem Cluster.

### 4.3.9. Datenträgergruppen entfernen

Verwenden Sie den Befehl **vgremove**, um eine Datenträgergruppe zu entfernen, die keine logischen Datenträger enthält.

```
# vgremove officevg
  Volume group "officevg" successfully removed
```

### 4.3.10. Aufteilen einer Datenträgergruppe

Verwenden Sie den Befehl **vgsplit**, um die physischen Datenträger einer Datenträgergruppe aufzuteilen und eine neue Datenträgergruppe zu erstellen.

Logische Datenträger können nicht zwischen Datenträgergruppen aufgeteilt werden. Jeder existierende logische Datenträger muss sich vollständig auf dem physischen Datenträger befinden und entweder die

alte oder die neue Datenträgergruppe bilden. Falls jedoch erforderlich können Sie die Aufteilung mithilfe des Befehls **pvmove** erzwingen.

Das folgende Beispiel trennt die neue Datenträgergruppe **smallvg** von der ursprünglichen Datenträgergruppe **bigvg**.

```
# vgsplit bigvg smallvg /dev/ram15
  Volume group "smallvg" successfully split from "bigvg"
```

### 4.3.11. Datenträgergruppen kombinieren

Verwenden Sie den Befehl **vgmerge**, um zwei Datenträgergruppen zu einer einzelnen Datenträgergruppe zu kombinieren. Sie können einen inaktiven Quell-Datenträger mit einem aktiven oder einem inaktiven Ziel-Datenträger vereinen, falls die Größe der physischen Extents des Datenträgers gleich ist und die Summe der physischen und logischen Datenträger beider Datenträgergruppen in die Beschränkung der Ziel-Datenträgergruppe passt.

Der folgende Befehl vereint die inaktive Datenträgergruppe **my_vg** mit der aktiven oder inaktiven Datenträgergruppe **databases** und gibt dabei umfangreiche Informationen zur Laufzeit aus.

```
vgmerge -v databases my_vg
```

### 4.3.12. Metadaten von Datenträgergruppen sichern

Sicherungskopien von Metadaten und Archiven werden automatisch bei jeder Änderung der Konfiguration von Datenträgergruppen und logischen Datenträgern erstellt, sofern dies nicht in der Datei **lvm.conf** deaktiviert wird. Standardmäßig wird das Backup der Metadaten in **/etc/lvm/backup** und das Backup der Metadaten-Archive in **/etc/lvm/archives** gespeichert. Sie können die Metadaten manuell mit dem Befehl **vgcfgbackup** in die Datei **/etc/lvm/backup** speichern.

Der Befehl **vgcfrestore** stellt die Metadaten einer Datenträgergruppe aus dem Archiv für alle physischen Datenträger in der Datenträgergruppe wieder her.

Werfen Sie einen Blick auf Abschnitt 6.4, „Wiederherstellen von Metadaten eines physischen Datenträgers“ für ein Beispiel zur Verwendung des Befehls **vgcfgrestore**, um Metadaten für physische Datenträger wiederherzustellen.

### 4.3.13. Datenträgergruppe umbenennen

Verwenden Sie den Befehl **vgrename**, um eine bestehende Datenträgergruppe umzubenennen.

Beide der folgenden Befehle benennen die bestehende Datenträgergruppe **vg02** in **my_volume_group** um.

```
vgrename /dev/vg02 /dev/my_volume_group
```

```
vgrename vg02 my_volume_group
```

### 4.3.14. Datenträgergruppe auf ein anderes System verschieben

Sie können eine komplette LVM-Datenträgergruppe auf ein anderes System portieren. Dabei wird empfohlen, dass Sie die Befehle **vgexport** und **vgimport** verwenden.

Mithilfe des Befehls **vgexport** wird eine inaktive Datenträgergruppe unzugänglich gemacht, so dass

Sie deren physische Datenträger entfernen können. Der Befehl **vgimport** macht eine Datenträgergruppe wieder für eine Maschine zugänglich, nachdem sie durch den Befehl **vgexport** auf inaktiv gesetzt wurde.

Um eine Datenträgergruppe von einem System auf ein anderes zu portieren, führen Sie die folgenden Schritte durch:

1. Stellen Sie sicher, dass keine Benutzer auf Dateien auf den aktiven Datenträgern in der Datenträgergruppe zugreifen und hängen anschließend die logischen Datenträger aus.
2. Verwenden Sie den Parameter **-a n** des Befehls **vgchange**, um die Datenträgergruppe als inaktiv zu markieren. Dies verhindert jegliche weitere Aktionen in der Datenträgergruppe.
3. Verwenden Sie den Befehl **vgexport**, um die Datenträgergruppe zu exportieren. Auf diese Weise wird der Zugriff auf sie von dem System aus, von dem Sie sie entfernen, verhindert.

   Nachdem Sie die Datenträgergruppe exportiert haben, erscheint der physische Datenträger als eine Datenträgergruppe, die exportiert wird, wenn Sie den Befehl **pvscan** ausführen, wie im folgenden Beispiel gezeigt.

   ```
   [root@tng3-1]# pvscan
     PV /dev/sda1    is in exported VG myvg [17.15 GB / 7.15 GB free]
     PV /dev/sdc1    is in exported VG myvg [17.15 GB / 15.15 GB free]
     PV /dev/sdd1   is in exported VG myvg [17.15 GB / 15.15 GB free]
     ...
   ```

   Beim nächsten Herunterfahren des Systems können Sie die Platten, aus denen die Datenträgergruppe besteht, entfernen und an das neue System hängen.
4. Wenn die Platten an das neue System gehängt werden, verwenden Sie den Befehl **vgimport**, um die Datenträgergruppe zu importieren und so den Zugriff auf sie für das neue System zu ermöglichen.
5. Aktivieren Sie die Datenträgergruppe mit dem Parameter **-a y** des Befehls **vgchange**.
6. Hängen Sie das Dateisystem ein, um es für die Verwendung freizugeben.

### 4.3.15. Verzeichnis für eine Datenträgergruppe neu erstellen

Verwenden Sie den Befehl **vgmknodes**, um ein Verzeichnis einer Datenträgergruppe und spezielle Dateien für logische Datenträger neu zu erstellen. Dieser Befehl überprüft die speziellen LVM2-Dateien im **/dev**-Verzeichnis, die für aktive logische Datenträger benötigt werden. Er erstellt alle speziellen Dateien, die fehlen, und entfernt nicht verwendete Dateien.

Sie können den **vgmknodes**-Befehl in den **vgscan**-Befehl mit einbinden, indem Sie den **mknodes**-Parameter zusammen mit dem **vgscan**-Befehl angeben.

## 4.4. Administration von logischen Datenträgern

Dieser Abschnitt beschreibt die Befehle, die die verschiedenen Aspekte der Administration von logischen Datenträgern abdecken.

### 4.4.1. Lineare logische Datenträger erstellen

Verwenden Sie den Befehl **lvcreate**, um einen logischen Datenträger zu erstellen. Falls Sie keinen Namen für den logischen Datenträger angeben, wird der standardmäßige Name **lvol#** verwendet, wobei # die interne Nummer des logischen Datenträgers darstellt.

Beim Erstellen eines logischen Datenträgers wird der logische Datenträger aus einer Datenträgergruppe

geschaffen, indem die freien Extents auf den physischen Datenträgern verwendet werden, aus denen die Datenträgergruppe besteht. Normalerweise verwenden logische Datenträger den nächstverfügbaren Platz auf dem zugrunde liegenden physischen Datenträgern. Das Ändern des logischen Datenträgers setzt Platz auf den physischen Datenträgern frei und weist diesen neu zu.

Der folgende Befehl erstellt einen logischen Datenträger mit der Größe von 10 Gigabytes in der Datenträgergruppe **vg1**.

```
lvcreate -L 10G vg1
```

Der folgende Befehl erstellt einen 1.500 Megabytes großen linearen logischen Datenträger namens **testlv** in der Datenträgergruppe **testvg** und erstellt dabei das Blockgerät **/dev/testvg/testlv**.

```
lvcreate -L1500 -n testlv testvg
```

Der folgende Befehl erstellt einen 50 Gigabyte großen logischen Datenträger namens **gfslv** aus den freien Extents in der Datenträgergruppe **vg0**.

```
lvcreate -L 50G -n gfslv vg0
```

Sie können den Parameter **-l** des **lvcreate**-Befehls verwenden, um die Größe des logischen Datenträgers in Extents anzugeben. Sie können diesen Parameter außerdem für die Festlegung verwenden, wie viel Prozent der Datenträgergruppe für den logischen Datenträger verwendet werden sollen. Der folgende Befehl erstellt einen logischen Datenträger namens **mylv**, der 60% des gesamten Platzes in der Datenträgergruppe **testvol** einnimmt.

```
lvcreate -l 60%VG -n mylv testvg
```

Sie können den Parameter **-l** des **lvcreate**-Befehls auch dazu verwenden, um den Prozentsatz des verbleibenden Platzes in einer Datenträgergruppe als Größe des logischen Datenträgers zu bestimmen. Der folgende Befehl erstellt einen logischen Datenträger namens **yourlv**, der den gesamten nicht zugewiesenen Platz in der Datenträgergruppe **testvol** verwendet.

```
lvcreate -l 100%FREE -n yourlv testvg
```

Sie können den Parameter **-l** des **lvcreate**-Befehls verwenden, um einen logischen Datenträger, der die gesamte logische Datenträgergruppe verwendet, zu erstellen. Eine weitere Möglichkeit, einen logischen Datenträger zu erstellen, der die gesamte Datenträgergruppe verwendet, ist die Verwendung des Befehls **vgdisplay**, um die "Total PE" Größe zu finden, und um diese Resultate als Eingabe für den Befehl **lvcreate** zu verwenden.

Die folgenden Befehle erstellen einen logischen Datenträger namens **mylv**, der die Datenträgergruppe namens **testvg** ausfüllt.

```
# vgdisplay testvg | grep "Total PE"
Total PE              10230
# lvcreate -l 10230 testvg -n mylv
```

Die zugrunde liegenden physischen Datenträger, die zur Erstellung eines logischen Datenträgers verwendet wurden, können von Bedeutung sein, wenn der physische Datenträger entfernt werden muss. Sie sollten diese Möglichkeit daher ggf. bei der Erstellung des logischen Datenträgers berücksichtigen. Werfen Sie einen Blick auf Abschnitt 4.3.6, „Physische Datenträger aus einer Datenträgergruppe entfernen“ für Informationen zum Entfernen eines physischen Datenträgers aus einer

Datenträgergruppe.

Um einen logischen Datenträger aus einem bestimmten physischen Datenträger in der Datenträgergruppe zu erstellen, fügen Sie den oder die physischen Datenträger am Ende der **lvcreate**-Befehlszeile ein. Der folgende Befehl erstellt einen logischen Datenträger namens **testlv** in der Datenträgergruppe **testvg**, zugewiesen vom physischen Datenträger **/dev/sdg1**.

```
lvcreate -L 1500 -ntestlv testvg /dev/sdg1
```

Sie können angeben, welche Extents eines physischen Datenträgers für einen logischen Datenträger verwendet werden sollen. Das folgende Beispiel erstellt einen linearen logischen Datenträger aus den Extents 0 bis 24 des physischen Datenträgers **/dev/sda1** und den Extents 50 bis 124 des physischen Datenträgers **/dev/sdb1** in der Datenträgergruppe **testvg**.

```
lvcreate -l 100 -n testlv testvg /dev/sda1:0-24 /dev/sdb1:50-124
```

Das folgende Beispiel erstellt einen linearen logischen Datenträger aus den Extents 0 bis 25 des physischen Datenträgers **/dev/sda1** und fährt mit der Anordnung des logischen Datenträgers bei Extent 100 fort.

```
lvcreate -l 100 -n testlv testvg /dev/sda1:0-25:100-
```

Die standardmäßige Richtlinie bezüglich der Anordnung eines logischen Datenträgers ist **inherit** (Vererbung) und wendet somit dieselbe Richtlinie an wie für die Datenträgergruppe. Diese Richtlinien können mithilfe des Befehls **lvchange** geändert werden. Werfen Sie einen Blick auf Abschnitt 4.3.1, „Datenträgergruppen erstellen“ für Informationen über Zuweisungsrichtlinien.

### 4.4.2. Striped-Datenträger erstellen

Bei großen sequentiellen Lese- und Schreibvorgängen kann das Erstellen eines logischen Striped-Datenträgers die Effizienz der Datenein- und -ausgabe verbessern. Werfen Sie einen Blick auf Abschnitt 2.3.2, „Logische Striped-Datenträger“ für allgemeine Informationen zu Striped-Datenträgern.

Beim Erstellen eines logischen Striped-Datenträgers geben Sie die Anzahl der Stripes mit dem Parameter **-i** des Befehls **lvcreate** an. Auf diese Weise wird festgelegt, über wie viele physischen Datenträger der logische Datenträger verteilt ("striped") wird. Die Anzahl der Stripes kann die Anzahl der physischen Datenträger in der Datenträgergruppe nicht übersteigen (sofern nicht der Parameter **--alloc anywhere** verwendet wird).

Falls die zugrunde liegenden physischen Geräte, aus dem ein logischer Striped-Datenträger besteht, unterschiedlich groß sind, wird die maximale Größe des Striped-Datenträgers vom kleinsten, zugrunde liegenden Gerät bestimmt. So entspricht beispielsweise die maximale Größe in einem Stripe mit zwei Standbeinen der zweifachen Größe des kleineren Geräts. In einem Stripe mit drei Standbeinen entspricht die maximale Größe der dreifachen Größe des kleinsten Geräts.

Der folgende Befehl erstellt einen logischen Striped-Datenträger über 2 physische Datenträger mit einem Stripe von 64 KB. Der logische Datenträger ist 50 Gigabytes groß, hat die Bezeichnung **gfslv** und wird aus der Datenträgergruppe **vg0** heraus erstellt.

```
lvcreate -L 50G -i2 -I64 -n gfslv vg0
```

Wie bei linearen Datenträgern können Sie die Extents des physischen Datenträgers angeben, den Sie für den Stripe verwenden. Der folgende Befehl erstellt einen Striped-Datenträger mit einer Größe von 100 Extents, der sich über zwei physische Datenträger erstreckt, die Bezeichnung **stripelv** trägt und

sich in der Datenträgergruppe **testvg** befindet. Der Stripe nutzt die Sektoren 0-49 von **/dev/sda1** und die Sektoren 50-99 von **/dev/sdb1**.

```
# lvcreate -l 100 -i2 -nstripelv testvg /dev/sda1:0-49 /dev/sdb1:50-99
  Using default stripesize 64.00 KB
  Logical volume "stripelv" created
```

### 4.4.3. Gespiegelte Datenträger erstellen



**Gespiegelte logische LVM-Datenträger in einem Cluster**

Zum Erstellen eines gespiegelten logischen LVM-Datenträgers in einem Cluster verwenden Sie dieselben Befehle und Prozeduren wie zum Erstellen eines gespiegelten logischen LVM-Datenträgers auf einem einzigen Knoten. Um jedoch einen gespiegelten logischen LVM-Datenträger in einem Cluster zu erzeugen, muss die Cluster-Infrastruktur ausgeführt werden und einsatzfähig sein, und der Sperrtyp in der **lvm.conf**-Datei muss richtig eingestellt sein, um Cluster-Sperren zu aktivieren. Ein Beispiel zur Erstellung eines gespiegelten Datenträgers in einem Cluster finden Sie in Abschnitt 5.5, „Erstellen eines gespiegelten logischen LVM-Datenträgers in einem Cluster“.
Wenn Sie versuchen, mehrere Befehle zur LVM-Mirror-Erstellung und -Konvertierung schnell hintereinander auszuführen, kann dies unter Umständen zu einem Rückstau dieser Befehle führen. Dadurch sind ggf. Zeitüberschreitungen für einige der angefragten Operationen die Folge, welche daraufhin fehlschlagen. Um dieses Problem zu vermeiden, empfehlen wir Ihnen, Befehle zur Cluster-Mirror-Erstellung von einem anderen Knoten im Cluster auszuführen.

Geben Sie beim Erstellen eines gespiegelten Datenträgers die Anzahl der Kopien, die von den Daten erstellt werden sollen mit dem Parameter **-m** des Befehls **lvcreate** an. Wird **-m1** angegeben, erstellt dies einen Mirror, der aus zwei Kopien des Dateisystems besteht: einem linearen logischen Datenträger plus einer Kopie. In gleicher Weise werden zwei Mirrors durch die Angabe von **-m2** erstellt, die aus drei Kopien des Dateisystems bestehen.

Der folgende Befehl erstellt einen gespiegelten logischen Datenträger mit einem einzelnen Mirror. Der Datenträger ist 50 Gigabytes groß, hat die Bezeichnung **mirrorlv** und wird aus der Datenträgergruppe **vg0** erstellt:

```
lvcreate -L 50G -m1 -n mirrorlv vg0
```

Ein LVM-Mirror unterteilt das Gerät, das kopiert wird, in Bereiche, die typischerweise 512 KB groß sind. Mithilfe des Parameters **-R** des **lvcreate**-Befehls können Sie die Bereichsgröße in MB spezifizieren. Sie können die Standardgröße für Bereiche ändern, indem Sie die **mirror_region_size**-Einstellung der **lvm.conf**-Datei ändern.



**Anmerkung**

Aufgrund von Einschränkungen in der Cluster-Infrastruktur können Cluster-Mirrors größer als 1.5 TB nicht mit der standardmäßigen Bereichsgröße von 512 KB erstellt werden. Benutzer, die größere Mirrors benötigen, sollten den Standardwert für die Bereichsgröße auf einen größeren Wert ändern. Wird die Bereichsgröße nicht angepasst, werden die LVM-Erstellung und ggf. andere LVM-Befehle hängen bleiben.
Als Faustregel zum Bestimmen der Bereichsgröße für Mirrors größer als 1,5 TB nehmen Sie Ihre Mirror-Größe in Terabytes, runden diesen Wert auf die nächste Zweierpotenz auf und verwenden diesen Wert als **-R**-Parameter zum **lvcreate**-Befehl. Ist Ihre Mirror-Größe beispielsweise 1,5 TB, können Sie **-R 2** angeben. Ist Ihre Mirror-Größe 3 TB, können Sie **-R 4** angeben. Für eine Mirror-Größe von 5 TB können Sie **-R 8** angeben.
Der folgende Befehl erzeugt einen gespiegelten logischen Datenträger mit einer Bereichsgröße von 2MB:

```
lvcreate -m1 -L 2T -R 2 -n mirror vol_group
```

LVM pflegt eine kleine Protokolldatei, in der festgehalten wird, welche Bereiche mit dem (den) Mirror(s) synchron sind. Standardmäßig wird diese Protokolldatei auf der Platte gespeichert, so dass sie über Neustarts hinaus bestehen bleibt. Alternativ können Sie angeben, dass diese Protokolldatei mit dem Parameter **--mirrorlog core** im Speicher behalten wird. Auf diese Weise wird kein extra Gerät zur Protokollierung benötigt, dies erfordert jedoch eine erneute Synchronisation des gesamten Mirrors bei jedem Neustart.

Der folgende Befehl erstellt einen gespiegelten logischen Datenträger aus der Datenträgergruppe **bigvg**. Der logische Datenträger hat die Bezeichnung **ondiskmirvol** und besitzt einen einzelnen Mirror. Er ist 12 MB groß und behält die Protokolldatei des Mirrors im Speicher.

```
# lvcreate -L 12MB -m1 --mirrorlog core -n ondiskmirvol bigvg
  Logical volume "ondiskmirvol" created
```

Das Mirror-Protokoll wird auf einem separaten Gerät erstellt, getrennt von den Mirrors selbst. Es ist jedoch möglich, das Mirror-Protokoll auf demselben Gerät anzulegen, auf dem sich auch eines der Mirror-Standbeine befindet, indem Sie den **--alloc anywhere**-Parameter des **vgcreate**-Befehls angeben. Dies kann sich nachteilig auf die Leistung auswirken, erlaubt Ihnen jedoch das Anlegen eines Mirrors, selbst wenn Sie nur über zwei zugrunde liegende Geräte verfügen.

Der folgende Befehl erstellt einen gespiegelten logischen Datenträger mit einem einzelnen Mirror, dessen Mirror-Protokoll sich auf demselben Gerät wie eines der Mirror-Standbeine befindet. In diesem Beispiel besteht die Datenträgergruppe **vg0** aus nur zwei Geräten. Dieser Befehl erstellt einen 500 Megabytes großen Datenträger namens **mirrorlv** in der **vg0** Datenträgergruppe.

```
lvcreate -L 500M -m1 -n mirrorlv -alloc anywhere vg0
```



**Anmerkung**

Bei geclusterten Mirrors ist der Cluster-Knoten mit der derzeit niedrigsten Cluster-ID für die Verwaltung des Mirror-Protokolls verantwortlich. Wenn das Gerät, welches das Cluster-Mirror-Protokoll enthält, auf einem Teilbereich des Clusters nicht verfügbar ist, kann der geclusterte Mirror somit ohne Einschränkungen weiterhin funktionieren, solange der Cluster-Knoten mit der niedrigsten ID weiterhin Zugriff auf das Mirror-Protokoll hat. Da der Mirror davon unberührt bleibt, wird auch keinerlei Aktion zur Korrektur (Reparation) veranlasst. Falls jedoch der Cluster-Knoten mit der niedrigsten ID den Zugriff auf das Mirror-Protokoll verliert, wird eine automatische Aktion ausgelöst (ungeachtet dessen, ob von anderen Knoten auf das Protokoll zugegriffen werden kann).

Um ein Mirror-Protokoll zu erstellen, das selbst wiederum gespiegelt ist, können Sie den **--mirrorlog mirrored**-Parameter angeben. Der folgende Befehl erstellt einen gespiegelten logischen Datenträger aus der Datenträgergruppe **bigvg**. Der logische Datenträger hat die Bezeichnung **twologvol** und besitzt einen einzelnen Mirror. Der Datenträger ist 12 MB groß, sein Mirror-Protokoll wird gespiegelt, und jede Protokolldatei wird auf einem separaten Gerät bewahrt.

```
# lvcreate -L 12MB -m1 --mirrorlog mirrored -n twologvol bigvg
  Logical volume "twologvol" created
```

Wie auch beim standardmäßigen Mirror-Protokoll ist es möglich, die redundanten Mirror-Protokolle auf demselben Gerät wie die Mirror-Standbeine zu platzieren, indem Sie den **--alloc anywhere**-Parameter des **vgcreate**-Befehls angeben. Dies kann sich nachteilig auf die Leistung auswirken, erlaubt Ihnen jedoch das Anlegen eines redundanten Mirror-Protokolls, selbst wenn Sie nicht über genügend zugrunde liegende Geräte verfügen, um jedes Protokoll auf einem separaten Gerät zu den Mirror-Standbeinen zu platzieren.

Beim Erstellen eines Mirrors werden die Abschnitte desselben synchronisiert. Bei großen Mirror-Komponenten kann der Synchronisationsprozess lange dauern. Wenn Sie einen neuen Mirror erstellen, der nicht erneuert werden muss, können Sie den Parameter **nosync** verwenden, um zu signalisieren, dass eine ursprüngliche Synchronisation vom ersten Gerät aus nicht erforderlich ist.

Sie können angeben, welche Geräte für die Protokolldatei des Mirrors und die Protokolldatei verwendet werden sollen und welche Extents der Geräte benutzt werden sollen. Um zu erzwingen, dass die Protokolldatei auf einer bestimmten Platte gespeichert werden soll, geben Sie genau einen Extent auf der Platte an, auf der diese platziert werden soll. LVM respektiert nicht unbedingt die Reihenfolge, in der Geräte auf der Befehlszeile aufgelistet sind. Falls ein beliebiger physischer Datenträger aufgelistet ist, ist dies der einzige Platz, auf dem die Zuweisung stattfindet. Jegliche physischen Extents in der Liste, die bereits zugewiesen sind, werden ignoriert.

Der folgende Befehl erstellt einen gespiegelten logischen Datenträger mit einem einzelnen Mirror und einem einzelnen, nicht gespiegelten Protokoll. Der Datenträger ist 500 Megabytes groß, hat die Bezeichnung **mirrorlv** und wird aus der Datenträgergruppe **vg0** heraus erstellt. Das erste Standbein des Mirrors liegt auf dem Gerät **/dev/sda1**, das zweite Standbein des Mirrors befindet sich auf dem Gerät **/dev/sdb1** und die Protokolldatei des Mirrors ist auf **/dev/sdc1**.

```
lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1 /dev/sdb1 /dev/sdc1
```

Der folgende Befehl erstellt einen gespiegelten logischen Datenträger mit einem einzelnen Mirror. Er ist 500 Megabytes groß, hat die Bezeichnung **mirrorlv** und wird aus der Datenträgergruppe **vg0** heraus erstellt. Das erste Standbein des Mirrors befindet sich auf den Extents 0 bis 499 auf dem Gerät

**/dev/sda1**, das zweite Standbein des Mirrors befindet sich auf den Extents 0 bis 499 auf dem Gerät **/dev/sdb1** und die Protokolldatei des Mirrors fängt auf Extent 0 des Geräts **/dev/sdc1** an. Letztere sind 1 MB Extents. Falls irgendwelche der angegebenen Extents bereits zugewiesen wurden, werden sie ignoriert.

```
lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1:0-499 /dev/sdb1:0-499 /dev/sdc1:0
```



**Anmerkung**

Ab der Red Hat Enterprise Linux 6.1 Release können Sie RAID0 (Striping) und RAID1 (Mirroring) auf einem einzigen logischen Datenträger kombinieren. Wenn Sie beim Erstellen eines logischen Datenträgers gleichzeitig die Anzahl der Mirrors (**--mirrors X**) sowie die Anzahl der Stripes (**--stripes Y**) angeben, wird dadurch ein Mirror-Gerät erstellt, dessen zugrunde liegenden Geräte gestriped sind.

### 4.4.3.1. Ausfallrichtlinie für gespiegelte logische Datenträger

Mithilfe der **mirror_image_fault_policy** und **mirror_log_fault_policy**-Parameter im **activation**-Abschnitt der **lvm.conf**-Datei können Sie definieren, wie ein gespiegelter logischer Datenträger sich bei einem Ausfall des Gerätes verhalten soll. Ist dieser Parameter auf **remove** gesetzt, versucht das System, das fehlerhafte Gerät zu entfernen und ohne es weiterzuarbeiten. Ist dieser Parameter auf **allocate** gesetzt, versucht das System, das fehlerhafte Gerät zu entfernen und Speicherplatz auf einem neuen Gerät als Ersatz für das ausgefallene Gerät zuzuweisen; diese Richtlinie verhält sich wie die **remove**-Richtlinie, falls kein passendes Gerät als Ersatz zugewiesen werden kann.

Standardmäßig ist der **mirror_log_fault_policy**-Parameter auf **allocate** festgelegt. Der Einsatz dieser Richtlinie für das Protokoll ist schnell und ermöglicht das Beibehalten des Sync-Zustands über Ausfälle und Neustarts hinweg. Wenn Sie diese Richtlinie auf **remove** setzen, konvertiert der Mirror beim Ausfall eines Protokollierungsgeräts zur Verwendung eines speicherinternen Protokolls und der Mirror behält seinen Sync-Zustand nicht über Ausfälle und Neustarts hinweg, so dass der gesamte Mirror neu synchronisiert werden muss.

Standardmäßig ist der **mirror_image_fault_policy**-Parameter auf **remove** festgelegt. Mit dieser Richtlinie wird der Mirror beim Ausfall eines Mirror-Images zu einem nicht-gespiegelten Gerät konvertiert, wenn es nur noch eine einzige intakte Kopie gibt. Setzen Sie diese Richtlinie auf **allocate** für ein Mirror-Gerät, verlangsamt dies den Prozess, da der Mirror die Geräte neu synchronisieren muss; doch die Mirror-Charakteristiken des Geräts bleiben erhalten.



**Anmerkung**

Wenn in einem LVM-Mirror ein Gerät ausfällt, erfolgt eine Wiederherstellung in zwei Stufen. In der ersten Stufe wird das ausgefallene Gerät entfernt. Dies kann dazu führen, das der Mirror zu einem linearen Gerät degradiert wird. Falls der **mirror_log_fault_policy**-Parameter auf **allocate** gesetzt ist, wird in der zweiten Stufe versucht, das ausgefallene Gerät zu ersetzen. Beachten Sie jedoch, dass es keine Garantie dafür gibt, dass in der zweiten Stufe Geräte gewählt werden, die bereits vom Mirror verwendet wurden, jedoch nicht vom Ausfall betroffen waren, wenn andere Geräte zur Verfügung stehen.
Informationen über die manuelle Wiederherstellung nach einem LVM-Mirror-Ausfall finden Sie in Abschnitt 6.3, „Wiederherstellung beim Ausfall eines LVM-Mirrors“.

#### 4.4.3.2. Abtrennen eines redundanten Images von einem gespiegelten logischen Datenträger

Sie können ein redundantes Image von einem gespiegelten logischen Datenträger abtrennen, um einen neuen logischen Datenträger zu formen. Um ein Image abzutrennen, verwenden Sie den **--splitmirrors**-Parameter des **lvconvert**-Befehls und geben die Anzahl der abzutrennenden Images an. Sie müssen den **--name**-Parameter des Befehls verwenden, um einen Namen für den neu abgetrennten logischen Datenträger zu spezifizieren.

Der folgende Befehl trennt einen neuen logischen Datenträger namens **copy** vom gespiegelten logischen Datenträger **vg/lv** ab. Der neue logische Datenträger enthält zwei Mirror-Standbeine. In diesem Beispiel entscheidet LVM, welches Gerät abgetrennt werden soll.

```
lvconvert --splitmirrors 2 --name copy vg/lv
```

Alternativ können Sie auch selbst angeben, welche Geräte abgetrennt werden sollen. Der folgende Befehl trennt einen neuen logischen Datenträger namens **copy** vom gespiegelten logischen Datenträger **vg/lv** ab. Der neue logische Datenträger enthält zwei Mirror-Standbeine, bestehend aus den Geräten **/dev/sdc1** und **/dev/sde1**.

```
lvconvert --splitmirrors 2 --name copy vg/lv /dev/sd[ce]1
```

#### 4.4.3.3. Gespiegelte Datenträger reparieren

Sie können den **lvconvert --repair**-Befehl verwenden, um einen Mirror nach einem Plattenausfall zu reparieren. Dies bringt den Mirror wieder auf einen konsistenten Zustand. Der **lvconvert --repair**-Befehl ist ein interaktiver Befehl, der Sie dazu auffordert anzugeben, ob das System versuchen soll, jegliche ausgefallenen Geräte zu ersetzen.

- Um die Eingabeaufforderungen zu überspringen und alle ausgefallenen Geräte zu ersetzen, geben Sie die **-y**-Option auf der Befehlszeile an.
- Um die Eingabeaufforderungen zu überspringen und keine der ausgefallenen Geräte zu ersetzen, geben Sie die **-f**-Option auf der Befehlszeile an.
- Um die Eingabeaufforderungen zu überspringen und dennoch verschiedene Richtlinien zur Platzierung des Mirror-Images und des Mirror-Protokolls zu spezifizieren, geben Sie den **--use-policies**-Parameter an, um die Richtlinien zum Geräteausfall zu verwenden, die von den **mirror_log_fault_policy**- und **mirror_device_fault_policy**-Parametern in der **lvm.conf**-Datei festgelegt sind.

#### 4.4.3.4. Konfigurationen von gespiegelten Datenträgern ändern

Sie können einen logischen Datenträger mit dem Befehl **lvconvert** von einem gespiegelten Datenträger in einen linearen Datenträger oder von einem linearen Datenträger in einen gespiegelten Datenträger konvertieren. Mit diesem Befehl können Sie auch anderer Parameter eines vorhandenen logischen Datenträgers neu konfigurieren, wie beispielsweise **corelog**.

Wenn Sie einen logischen Datenträger in einen gespiegelten Datenträger konvertieren, erstellen Sie im Wesentlichen Standbeine für einen Mirror für einen vorhandenen Datenträger. Dies bedeutet, dass Ihre Datenträgergruppe die Geräte und den Platz für die Standbeine des Mirrors und für die Protokolldatei des Mirrors umfassen muss.

Falls Sie ein Standbein des Mirrors verlieren, konvertiert LVM den Datenträger in einen linearen Datenträger, so dass Sie immer noch Zugriff auf den Datenträger besitzen, ohne die Redundanz des Mirrors. Nachdem Sie das Standbein ersetzt haben, können Sie den Mirror mithilfe des Befehls

**lvconvert** wiederherstellen. Diese Prozedur wird in Abschnitt 6.3, „Wiederherstellung beim Ausfall eines LVM-Mirrors“ erläutert.

Der folgende Befehl konvertiert den logischen linearen Datenträger **vg00/lvol1** in einen gespiegelten logischen Datenträger.

```
lvconvert -m1 vg00/lvol1
```

Der folgende Befehl konvertiert den gespiegelten logischen Datenträger **vg00/lvol1** in einen logischen linearen Datenträger und entfernt das Standbein des Mirrors.

```
lvconvert -m0 vg00/lvol1
```

### 4.4.4. Snapshot-Datenträger erstellen

Verwenden Sie den Parameter **-s** des Befehls **lvcreate**, um einen Snapshot-Datenträger zu erstellen. Ein Snapshot-Datenträger ist beschreibbar.



**Anmerkung**

LVM-Snapshots werden nicht über Knoten im Cluster hinweg unterstützt. Sie können keinen Snapshot-Datenträger in einer geclusterten Datenträgergruppe erstellen. Ab der Red Hat Enterprise Linux 6.1 Release können Sie jedoch, um eine konsistente Datensicherung eines geclusterten logischen Datenträgers durchzuführen, den Datenträger exklusiv aktivieren und dann den Snapshot erstellen. Weitere Informationen über das Aktivieren von logischen Datenträgern exklusiv auf einem Knoten finden Sie in Abschnitt 4.7, „Logische Datenträger auf einzelnen Knoten in einem Cluster aktivieren“.



**Anmerkung**

Ab der Red Hat Enterprise Linux 6.1 Release werden LVM-Snapshots nunmehr für gespiegelte logische Datenträger unterstützt.

Der folgende Befehl erstellt einen logischen Snapshot-Datenträger, der 100 Megabytes groß ist und die Bezeichnung **/dev/vg00/snap** trägt. Dies erstellt einen Snapshot des ursprünglichen logischen Datenträgers mit der Bezeichnung **/dev/vg00/lvol1**. Falls der ursprüngliche logische Datenträger ein Dateisystem besitzt, können Sie den logischen Snapshot-Datenträger auf einem beliebigen Verzeichnis einhängen, um auf den Inhalt des Dateisystems zuzugreifen und eine Sicherung durchzuführen, während das ursprüngliche Dateisystem weiterhin aktualisiert wird.

```
lvcreate --size 100M --snapshot --name snap /dev/vg00/lvol1
```

Nach der Erstellung eines logischen Snapshot-Datenträgers liefert der Befehl **lvdisplay** auf den ursprünglichen logischen Datenträgern eine Ausgabe, die eine Liste aller logischen Snapshot-Datenträger und ihrem Status (aktiv oder inaktiv) enthält.

Das folgende Beispiel zeigt den Status des logischen Datenträgers **/dev/new_vg/lvol0**, für den der Snapshot-Datenträger **/dev/new_vg/newvgsnap** erstellt wurde.

```
# lvdisplay /dev/new_vg/lvol0
  --- Logical volume ---
  LV Name                /dev/new_vg/lvol0
  VG Name                new_vg
  LV UUID                LBy1Tz-sr23-OjsI-LT03-nHLC-y8XW-EhCl78
  LV Write Access        read/write
  LV snapshot status     source of
                         /dev/new_vg/newvgsnap1 [active]
  LV Status              available
  # open                 0
  LV Size                52.00 MB
  Current LE             13
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:2
```

Der Befehl **lvs** zeigt standardmäßig den Ursprungsdatenträger und den aktuellen Prozentsatz des Snapshot-Datenträgers und wird für jeden Snapshot-Datenträger benutzt. Das folgende Beispiel zeigt die standardmäßige Ausgabe des Befehls **lvs** für ein System, dass den logischen Datenträger **/dev/new_vg/lvol0** umfasst, für den der Snapshot-Datenträger **/dev/new_vg/newvgsnap** erstellt wurde.

```
# lvs
  LV         VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0      new_vg owi-a- 52.00M
  newvgsnap1 new_vg swi-a-  8.00M lvol0    0.20
```



**Anmerkung**

Da sich der Snapshot vergrößert, wenn sich der ursprüngliche Datenträger ändert, ist es wichtig, den Prozentsatz des Snapshot-Datenträgers regelmäßig mit dem Befehl **lvs** zu überwachen um sicherzustellen, dass er sich nicht auffüllt. Ein Snapshot, der 100% voll ist, ist komplett verloren, da Schreibversuche auf unveränderte Teile des Quelldatenträgers nicht möglich sind, ohne den Snapshot zu korrumpieren.

### 4.4.5. Snapshot-Datenträger zusammenführen

Ab der Red Hat Enterprise Linux 6 Release können Sie die **--merge**-Option des **lvconvert**-Befehls verwenden, um einen Snapshot wieder mit dem ursprünglichen Datenträger zusammenzuführen. Wenn weder der ursprüngliche noch der Snapshot-Datenträger geöffnet sind, beginnt die Zusammenführung sofort. Andernfalls beginnt die Zusammenführung, sobald entweder das Original oder der Snapshot aktiviert werden und beide geschlossen sind. Das Zusammenführen eines Snapshots mit einem Original, das nicht geschlossen werden kann, wie z.B. einem Root-Dateisystem, wird verschoben, bis der originale Datenträger das nächste mal aktiviert wird. Beim Zusammenführen erhält der daraus entstehende logische Datenträger den Namen, die Minor-Nummer und die UUID des originalen Datenträgers. Während der Zusammenführung erscheinen Lese- oder Schreibvorgänge auf dem Original, als wenn diese zum Snapshot umgeleitet werden. Nach Abschluss der Zusammenführung wird der Snapshot gelöscht.

Der folgende Befehl führt den Snapshot-Datenträger **vg00/lvol1_snap** mit seinem Original zusammen.

```
lvconvert --merge vg00/lvol1_snap"
```

Sie können in der Befehlszeile mehrere Snapshots angeben, oder Sie können LVM-Objekt-Tags verwenden, um mehrere Snapshots anzugeben, die mit ihren jeweiligen Originaldatenträgern zusammengeführt werden sollen. In dem folgenden Beispiel haben die logischen Datenträger **vg00/lvol1**, **vg00/lvol2** und **vg00/lvol3** alle den Tag **@some_tag**. Der folgende Befehl führt diese logischen Snapshot-Datenträger der Reihe nach für alle drei Datenträger zusammen: **vg00/lvol1**, dann **vg00/lvol2**, dann **vg00/lvol3**. Wäre die **--background**-Option angegeben, würden alle Zusammenführungen der logischen Datenträger gleichzeitig starten.

```
lvconvert --merge @some_tag"
```

Informationen über das Tagging von LVM-Objekten finden Sie in Anhang C, *LVM Objekt-Tags*. Werfen Sie einen Blick auf die Handbuchseite von **lvconvert**(8) für weitere Informationen über den **lvconvert --merge**-Befehl.

### 4.4.6. Persistente Gerätenummern

Major- und Minor-Gerätenummern werden dynamisch beim Laden der Module zugewiesen. Einige Applikationen funktionieren am besten, wenn das Blockgerät immer mit der gleichen Gerätenummer (Major und Minor) aktiviert wird. Diese können Sie mit den Befehlen **lvcreate** und **lvchange** angeben und folgende Parameter benutzen:

```
--persistent y --major major --minor minor
```

Verwenden Sie eine große Minor-Nummer, um sicherzustellen, dass diese nicht bereits einem anderen Gerät dynamisch zugewiesen wurde.

Falls Sie ein Dateisystem via NFS exportieren, kann die Angabe des Parameters **fsid** in der exports-Datei das Setzen einer persistenten Gerätenummer innerhalb von LVM überflüssig machen.

### 4.4.7. Größe von Logischen Datenträger anpassen

Verwenden Sie den Befehl **lvreduce**, um die Größe eines logischen Datenträgers zu ändern. Ist auf dem logischen Datenträger ein Dateisystem vorhanden, stellen Sie sicher, dass das Dateisystem zuerst verkleinert wird (Sie können auch das LVM-GUI verwenden), so dass der logische Datenträger immer mindestens so groß ist, wie das Dateisystem es erwartet.

Der folgende Befehl reduziert die Größe des logischen Datenträgers **lvol1** in der Datenträgergruppe **vg00** um 3 logische Extents.

```
lvreduce -l -3 vg00/lvol1
```

### 4.4.8. Parameter einer logischen Datenträgergruppe ändern

Verwenden Sie den Befehl **lvchange**, um die Parameter eines logischen Datenträgers zu verändern. Werfen Sie einen Blick auf die Handbuchseite (8) des Befehls **lvchange** für eine Auflistung der Parameter, die Sie ändern können.

Mithilfe des Befehls **lvchange** können Sie logische Datenträger aktivieren und deaktivieren. Um alle logischen Datenträger in einer Datenträgergruppe gleichzeitig zu aktivieren bzw. zu deaktivieren, verwenden Sie den Befehl **vgchange**, wie in Abschnitt 4.3.7, „Parameter einer Datenträgergruppe verändern“ beschrieben.

Der folgende Befehl ändert die Zugriffsrechte auf dem Datenträger **lvol1** in der Datenträgergruppe **vg00** in schreibgeschützt.

```
lvchange -pr vg00/lvol1
```

### 4.4.9. Logische Datenträger umbenennen

Verwenden Sie den Befehl **lvrename**, um einen bestehenden logischen Datenträger umzubenennen.

Einer der folgenden Befehle benennt den logischen Datenträger **lvold** in der Datenträgergruppe **lvold** in **lvnew** um.

```
lvrename /dev/vg02/lvold /dev/vg02/lvnew
```

```
lvrename vg02 lvold lvnew
```

Werfen Sie einen Blick auf Abschnitt 4.7, „Logische Datenträger auf einzelnen Knoten in einem Cluster aktivieren“ für weitere Informationen zur Aktivierung logischer Datenträger auf einzelnen Knoten in einem Cluster.

### 4.4.10. Logische Datenträger entfernen

Verwenden Sie den Befehl **lvremove**, um inaktive logische Datenträger zu entfernen. Bevor ein logischer Datenträger entfernt werden kann, muss er ausgehängt werden. In einer Cluster-Umgebung muss ein logischer Datenträger zusätzlich noch deaktiviert werden, bevor er entfernt werden kann.

Der folgende Befehl entfernt den logischen Datenträger **/dev/testvg/testlv** aus der Datenträgergruppe **testvg**. Beachten Sie, dass in diesem Fall der logische Datenträger nicht deaktiviert wurde.

```
[root@tng3-1 lvm]# lvremove /dev/testvg/testlv
Do you really want to remove active logical volume "testlv"? [y/n]: y
  Logical volume "testlv" successfully removed
```

Sie können den logischen Datenträger explizit mit dem Befehl **lvchange -an** deaktivieren, bevor sie ihn entfernen, woraufhin Sie den Prompt, der verifiziert, ob Sie einen aktiven logischen Datenträger entfernen möchten, nicht sehen werden.

### 4.4.11. Logische Datenträger anzeigen

Es gibt drei Befehle, mithilfe derer Sie sich die Eigenschaften von logischen LVM-Datenträgern anzeigen lassen können: **lvs**, **lvdisplay** und **lvscan**.

Der Befehl **lvs** liefert Informationen zu einem logischen Datenträger in einer konfigurierbaren Form, wobei eine Zeile pro logischem Datenträger angezeigt wird. Der **lvs**-Befehl bietet ein hohes Maß an Formatkontrolle und ist nützlich für das Skripting. Werfen Sie einen Blick auf Abschnitt 4.8, „Angepasste Berichterstattung für LVM“ für Informationen zur Verwendung des Befehls **lvs** zur Anpassung Ihrer Ausgabe.

Der Befehl **lvdisplay** zeigt Eigenschaften (wie Größe, Layout und Mapping) in einem festen Format an.

Der folgende Befehl zeigt die Attribute von **lvol2** in **vg00** an. Falls logische Snapshot-Datenträger für diesen originalen logischen Datenträger erstellt wurden, zeigt dieser Befehl auch eine Liste aller

logischen Snapshot-Datenträger und ihren Status (aktiv oder inaktiv) an.

```
lvdisplay -v /dev/vg00/lvol2
```

Der Befehl **lvscan** sucht das System nach allen logischen Datenträgern ab und listet diese auf, wie in folgendem Beispiel.

```
# lvscan
 ACTIVE                   '/dev/vg0/gfslv' [1.46 GB] inherit
```

## 4.4.12. Logische Datenträger vergrößern

Verwenden Sie den Befehl **lvextend**, um die Größe eines logischen Datenträgers zu erweitern.

Bei der Vergrößerung des logischen Datenträgers können Sie angeben, um wie viel Sie den Datenträger erweitern möchten, oder wie groß dieser nach der Erweiterung sein soll.

Der folgende Befehl vergrößert den logischen Datenträger **/dev/myvg/homevol** auf 12 Gigabytes.

```
# lvextend -L12G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 12 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

Der folgende Befehl fügt ein zusätzliches Gigabyte zum logischen Datenträger **/dev/myvg/homevol** hinzu.

```
# lvextend -L+1G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 13 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

Wie beim Befehl **lvcreate** können Sie mithilfe des Parameters **-l** des Befehls **lvextend** die Anzahl der Extents angeben, um die der logische Datenträger vergrößert werden soll. Sie können diesen Parameter auch dazu verwenden, um einen Prozentsatz der Datenträgergruppe oder einen Prozentsatz des verbleibenden freien Platzes in der Datenträgergruppe anzugeben. Der folgende Befehl erweitert den logischen Datenträger mit der Bezeichnung **testlv** so, dass der gesamte nicht zugewiesene Platz in der Datenträgergruppe **myvg** aufgefüllt wird.

```
[root@tng3-1 ~]# lvextend -l +100%FREE /dev/myvg/testlv
  Extending logical volume testlv to 68.59 GB
  Logical volume testlv successfully resized
```

Nach der Erweiterung des logischen Datenträgers müssen Sie die Größe des dazugehörigen Dateisystems so vergrößern, dass dieses übereinstimmt.

Standardmäßig vergrößern die meisten Tools zur Einteilung der Größe eines Dateisystems dieses so, dass es mit der Größe des zugrunde liegenden logischen Datenträgers übereinstimmt, so dass Sie sich nicht darum kümmern müssen, dieselbe Größe für jeden der beiden Befehle anzugeben.

### 4.4.12.1. Striped-Datenträger vergrößern

Um die Größe eines logischen Striped-Datenträgers zu erhöhen, muss genug freier Platz auf dem zugrunde liegenden physischen Datenträger vorhanden sein, aus dem die Datenträgergruppe besteht, um den Stripe zu unterstützen. Falls Sie beispielsweise einen zweigleisigen Stripe besitzen, der eine

gesamte Datenträgergruppe beansprucht, führt das Hinzufügen eines einzelnen physischen Datenträgers zu der Datenträgergruppe nicht zu einer Erweiterung des Stripes. Sie müssen stattdessen mindestens zwei physische Datenträger zur Datenträgergruppe hinzufügen.

Stellen Sie sich beispielsweise eine Datenträgergruppe **vg** vor, die aus zwei zugrunde liegenden physischen Datenträgern besteht, wie mit dem folgenden Befehl **vgs** dargestellt.

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   0   0 wz--n- 271.31G 271.31G
```

Sie können einen Stripe erstellen und dabei den gesamten Platz in der Datenträgergruppe verwenden.

```
# lvcreate -n stripe1 -L 271.31G -i 2 vg
  Using default stripesize 64.00 KB
  Rounding up size to full physical extent 271.31 GB
  Logical volume "stripe1" created
# lvs -a -o +devices
  LV      VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe1 vg   -wi-a- 271.31G
/dev/sda1(0),/dev/sdb1(0)
```

Beachten Sie, dass die Datenträgergruppe keinen freien Platz mehr besitzt.

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   1   0 wz--n- 271.31G    0
```

Der folgende Befehl fügt einen weiteren physischen Datenträger zu der Datenträgergruppe hinzu, welche dann über 135 GB zusätzlichen Platz verfügt.

```
# vgextend vg /dev/sdc1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     3   1   0 wz--n- 406.97G 135.66G
```

Zu diesem Zeitpunkt können Sie den logischen Striped-Datenträger nicht auf die komplette Größe der Datenträgergruppe ausweiten, da zwei zugrunde liegende Geräte benötigt werden, um die Daten zu verteilen.

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
  Insufficient suitable allocatable extents for logical volume stripe1: 34480
more required
```

Fügen Sie einen weiteren physischen Datenträger hinzu, um den logischen Striped-Datenträger zu vergrößern und erweitern anschließend den logischen Datenträger. Nachdem wir in diesem Beispiel zwei physische Datenträger zu der Datenträgergruppe hinzugefügt haben, können wir den logischen Datenträger auf die gesamte Größe der Datenträgergruppe ausweiten.

```
# vgextend vg /dev/sdd1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     4   1   0 wz--n- 542.62G 271.31G
# lvextend vg/stripe1 -L 542G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 542.00 GB
  Logical volume stripe1 successfully resized
```

Falls Sie nicht genügend zugrunde liegende physische Geräte besitzen, um den logischen Striped-Datenträger zu erweitern, kann der Datenträger trotzdem erweitert werden, falls es keine Rolle spielt, dass die Erweiterung nicht verteilt (striped) wird. Beim Hinzufügen von Platz zum logischen Datenträger werden gemäß der standardmäßigen Vorgehensweise dieselben Striping-Parameter des letzten Segments des vorhandenen logischen Datenträgers verwendet. Diese Parameter können jedoch außer Kraft gesetzt werden. Das folgende Beispiel erweitert den existierenden logischen Striped-Datenträger, so dass dieser den verbleibenden Platz verwendet, nachdem der ursprüngliche Befehl **lvextend** fehlschlägt.

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
  Insufficient suitable allocatable extents for logical volume stripe1: 34480
more required
# lvextend -i1 -l+100%FREE vg/stripe1
```

### 4.4.12.2. Erweitern eines logischen Datenträgers mit der cling-Zuweisungsrichtlinie

Beim Erweitern eines LVM-Datenträgers können Sie die **--alloc cling**-Option des **lvextend**-Befehls verwenden, um die **cling**-Zuweisungsrichtlinie zu spezifizieren. Diese Richtlinie wählt Speicherplatz auf denselben physischen Datenträgern, auf denen sich das letzte Segment des vorhandenen logischen Datenträgers befindet. Falls die physischen Datenträger nicht ausreichend Platz bieten und eine Liste mit Tags in der **lvm.conf**-Datei definiert ist, überprüft LVM, ob diese Tags mit den physischen Datenträgern verknüpft sind und versucht, diese physischen Datenträger-Tags zwischen den vorhandenen und den neuen Extents abzugleichen.

Wenn Sie beispielsweise logische Datenträger, die an zwei Standorten gespiegelt sind, innerhalb derselben Datenträgergruppe haben, können Sie die physischen Datenträger je nach Standort mit Tags versehen, indem Sie den physischen Datenträgern die Tags @site1 und @site2 zuweisen und die folgende Zeile in der **lvm.conf**-Datei einfügen:

```
cling_tag_list = [ "@site1", "@site2" ]
```

Werfen Sie einen Blick auf *Anhang C, LVM Objekt-Tags* für Informationen zum Tagging von physischen Datenträgern.

In dem folgenden Beispiel wurde die **lvm.conf**-Datei verändert und enthält nun die folgende Zeile:

```
cling_tag_list = [ "@A", "@B" ]
```

Zudem wurde in diesem Beispiel eine Datenträgergruppe namens **taft** erstellt, welche die folgenden physischen Datenträger umfasst: **/dev/sdb1**, **/dev/sdc1**, **/dev/sdd1**, **/dev/sde1**, **/dev/sdf1**, **/dev/sdg1** und **/dev/sdh1**. Diesen physischen Datenträgern wurden die Tags **A**, **B** und **C** zugewiesen. Das Beispiel verwendet den **C**-Tag zwar nicht, doch veranschaulicht dies, dass LVM die

Tags nutzt, um die physischen Datenträger für die Mirror-Standbeine auszuwählen.

```
[root@taft-03 ~]# pvs -a -o +pv_tags /dev/sd[bcdefgh]1
  PV         VG   Fmt  Attr PSize   PFree   PV Tags
  /dev/sdb1  taft lvm2 a-   135.66g 135.66g A
  /dev/sdc1  taft lvm2 a-   135.66g 135.66g B
  /dev/sdd1  taft lvm2 a-   135.66g 135.66g B
  /dev/sde1  taft lvm2 a-   135.66g 135.66g C
  /dev/sdf1  taft lvm2 a-   135.66g 135.66g C
  /dev/sdg1  taft lvm2 a-   135.66g 135.66g A
  /dev/sdh1  taft lvm2 a-   135.66g 135.66g A
```

Der folgende Befehl erstellt einen 100 GB großen gespiegelten Datenträger in der Datenträgergruppe **taft**.

```
[root@taft-03 ~]# lvcreate -m 1 -n mirror --nosync -L 100G taft
```

Der folgende Befehl zeigt, welche Geräte für die Mirror-Standbeine und das Mirror-Protokoll verwendet werden.

```
[root@taft-03 ~]# lvs -a -o +devices
  LV                  VG         Attr   LSize   Log         Copy%  Devices
  mirror              taft       Mwi-a- 100.00g mirror_mlog 100.00
mirror_mimage_0(0),mirror_mimage_1(0)
  [mirror_mimage_0] taft       iwi-ao 100.00g                      /dev/sdb1(0)
  [mirror_mimage_1] taft       iwi-ao 100.00g                      /dev/sdc1(0)
  [mirror_mlog]     taft       lwi-ao   4.00m                      /dev/sdh1(0)
```

Der folgende Befehl erweitert die Größe des gespiegelten Datenträgers, wobei die **cling**-Zuweisungsrichtlinie verwendet wird um anzugeben, dass die Mirror-Standbeine unter Verwendung eines physischen Datenträgers mit demselben Tag erweitert werden sollen.

```
[root@taft-03 ~]# lvextend --alloc cling -L +100G taft/mirror
  Extending 2 mirror images.
  Extending logical volume mirror to 200.00 GiB
  Logical volume mirror successfully resized
```

Der folgende Befehl zeigt, dass die Mirror-Standbeine erweitert wurden unter Verwendung von physischen Datenträgern mit demselben Tag wie das Standbein. Beachten Sie, dass die physischen Datenträger mit dem Tag **C** ignoriert wurden.

```
[root@taft-03 ~]# lvs -a -o +devices
  LV                  VG         Attr   LSize   Log         Copy%  Devices
  mirror              taft       Mwi-a- 200.00g mirror_mlog  50.16
mirror_mimage_0(0),mirror_mimage_1(0)
  [mirror_mimage_0] taft       Iwi-ao 200.00g                      /dev/sdb1(0)
  [mirror_mimage_0] taft       Iwi-ao 200.00g                      /dev/sdg1(0)
  [mirror_mimage_1] taft       Iwi-ao 200.00g                      /dev/sdc1(0)
  [mirror_mimage_1] taft       Iwi-ao 200.00g                      /dev/sdd1(0)
  [mirror_mlog]     taft       lwi-ao   4.00m                      /dev/sdh1(0)
```

### 4.4.13. Logische Datenträger verkleinern

Um die Größe von logischen Datenträgern zu reduzieren, hängen Sie als Erstes das Dateisystem aus. Mithilfe des Befehls **lvreduce** können Sie dann den Datenträger verkleinern. Hängen Sie das Dateisystem nach der Verkleinerung des Datenträgers wieder ein.



**Warnung**

Es ist wichtig, dass die Größe des Dateisystems, bzw. was auch immer sich auf dem Datenträger befindet, zu verringern, bevor der Datenträger selbst verkleinert wird. Ansonsten riskieren Sie den Verlust von Daten.

Durch die Verkleinerung eines logischen Datenträgers wird ein Teil der Datenträgergruppe freigesetzt, der dann anderen logischen Datenträgern in der Datenträgergruppe zugewiesen werden sollte.

Das folgende Beispiel reduziert die Größe des logischen Datenträgers **lvol1** in der Datenträgergruppe **vg00** um 3 logische Extents.

```
lvreduce -l -3 vg00/lvol1
```

## 4.5. LVM-Geräte-Scans mit Filtern kontrollieren

Bei Systemstart wird der Befehl **vgscan** ausgeführt, um die Blockgeräte auf dem System nach LVM-Labels abzusuchen, und um daraufhin zu ermitteln, welche davon physische Datenträger sind und schließlich die Metadaten zu lesen und eine Liste von Datenträgergruppen zu erstellen. Die Namen der physischen Datenträger werden in der Cache-Datei **/etc/lvm/.cache** von jedem Knoten im System gespeichert. Nachfolgende Befehle können diese Datei lesen, um ein erneutes Scannen zu vermeiden.

Indem Sie Filter in der Konfigurationsdatei **lvm.conf** einrichten, können Sie kontrollieren, welche Geräte LVM absucht. Die Filter in der **lvm.conf**-Datei bestehen aus einer Reihe einfacher regulärer Ausdrücke, die auf die Gerätenamen im **/dev**-Verzeichnis anwendet werden um zu entscheiden, ob das jeweils gefundene Blockgerät akzeptiert oder zurückgewiesen werden soll.

Die folgenden Beispiele zeigen die Verwendung von Filtern zur Kontrolle, welche Geräte LVM absucht. Bitte beachten Sie, dass einige dieser Beispiele nicht unbedingt das optimale Verfahren darstellen, da die regulären Ausdrücke frei mit dem kompletten Pfadnamen verglichen werden. So entspricht **a/loop/** beispielsweise **a/.*loop.*/** und würde auf **/dev/solooperation/lvol1** zutreffen.

Der folgende Filter fügt alle entdeckten Geräte hinzu. Dies ist das Standardverhalten, da es keinen vorkonfigurierten Filter in der Konfigurationsdatei gibt:

```
filter = [ "a/.*/" ]
```

Der folgende Filter entfernt das CD-ROM-Gerät, um Verzögerungen zu vermeiden, falls das Laufwerk kein Medium enthält:

```
filter = [ "r|/dev/cdrom|" ]
```

Der folgende Filter fügt alle loop-Geräte hinzu und entfernt andere Blockgeräte:

```
filter = [ "a/loop.*/", "r/.*/" ]
```

Der folgende Filter fügt alle loop- und IDE-Geräte hinzu und entfernt alle anderen Blockgeräte:

```
filter =[ "a|loop.*|", "a|/dev/hd.*|", "r|.*|" ]
```

Der folgende Filter fügt lediglich Partition 8 auf dem ersten IDE-Laufwerk hinzu und entfernt alle anderen Blockgeräte:

```
filter = [ "a|^/dev/hda8$|", "r/.*/" ]
```

Werfen Sie einen Blick auf die Anhang B, *Die LVM-Konfigurationsdateien* und die Handbuchseite (5) von **lvm.conf** für weitere Informationen zur **lvm.conf**-Datei.

## 4.6. Online-Datenumzug

Mit dem Befehl **pvmove** können Sie im laufenden Betrieb des Systems Daten verschieben.

Der Befehl **pvmove** teilt die Daten so auf, dass sie in Abschnitte verteilt werden und erstellt einen temporären Mirror, um jeden Abschnitt zu verschieben. Werfen Sie einen Blick auf die Handbuchseite (8) von **pvmove** für weitere Informationen zur Anwendung von **pvmove**.

Der folgende Befehl verschiebt den gesamten zugewiesenen Platz vom physischen Datenträger **/dev/sdc1** auf andere freie physische Datenträger in der Datenträgergruppe:

```
pvmove /dev/sdc1
```

Der folgende Befehl verschiebt lediglich die Extents des logischen Datenträgers **MyLV**.

```
pvmove -n MyLV /dev/sdc1
```

Da die Ausführung des Befehls **pvmove** lange dauern kann, sollten Sie diesen ggf. im Hintergrund ausführen, um die Aktualisierung der Fortschrittsanzeige im Vordergrund zu vermeiden. Der folgende Befehl verschiebt alle für den physischen Datenträger **/dev/sdc1** zugewiesenen Extents im Hintergrund nach **/dev/sdf1**.

```
pvmove -b /dev/sdc1 /dev/sdf1
```

Der folgende Befehl gibt in 5-Sekunden Intervallen den Fortschritt der Verschiebung in Prozent wieder.

```
pvmove -i5 /dev/sdd1
```

## 4.7. Logische Datenträger auf einzelnen Knoten in einem Cluster aktivieren

Falls Sie LVM in einer Cluster-Umgebung installiert haben, müssen Sie gelegentlich logische Datenträger exklusiv auf einem Knoten aktivieren.

Verwenden Sie den Befehl **lvchange -aey**, um einen logischen Datenträger exklusiv auf einem Knoten zu aktivieren. Alternativ können Sie **lvchange -aly** verwenden, um logische Datenträger nur auf dem lokalen Knoten zu aktivieren, jedoch nicht ausschließlich. Sie können diese später gleichzeitig auf zusätzlichen Knoten aktivieren.

Sie können logische Datenträger auf einzelnen Knoten auch mit LVM-Tags aktivieren, wie in Anhang C, *LVM Objekt-Tags* beschrieben. Weiterhin können Sie die Aktivierung von Knoten in der Konfigurationsdatei angeben, was in Anhang B, *Die LVM-Konfigurationsdateien* beschrieben wird.

## 4.8. Angepasste Berichterstattung für LVM

Mit den Befehlen **pvs**, **lvs**, und **vgs** können Sie präzise und anpassbare Berichte von LVM-Objekten erstellen. Die von diesen Befehlen erstellten Berichte umfassen eine Ausgabezeile für jedes Objekt. Jede Zeile enthält eine sortierte Felderliste von Eigenschaften dieses Objekts. Es gibt fünf Möglichkeiten zur Auswahl der anzuzeigenden Objekte: pro physischem Datenträger, pro Datenträgergruppe, pro logischem Datenträger, pro physischem Datenträgersegment oder pro logischem Datenträgersegment.

Der folgende Abschnitt bietet:

- Eine Zusammenfassung der Befehlsparameter, die Sie zur Formatkontrolle des erstellten Berichts verwenden können.
- Eine Liste der Felder, die Sie für jedes LVM-Objekt auswählen können.
- Eine Zusammenfassung der Befehlsparameter, mit denen Sie den erstellten Bericht sortieren können.
- Anweisungen zur Angabe der Einheiten des Berichts.

### 4.8.1. Formatkontrolle

Abhängig davon, ob Sie die Befehle **pvs**, **lvs**, oder **vgs** verwenden, beeinflusst dies die standardmäßig angezeigten Felder und die Sortierungsreihenfolge. Sie können die Ausgabe dieser Befehle mit den folgenden Parametern steuern:

- Mithilfe des Parameters **-o** können Sie ändern, welche Felder zusätzlich zu den Standardfeldern angezeigt werden sollen. Nachfolgende Ausgabe ist die Standardausgabe für den Befehl **pvs** (der Informationen über physische Datenträger anzeigt).

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G
```

  Mit dem folgenden Befehl können Sie nur den Namen und die Größe eines physischen Datenträgers anzeigen.

```
# pvs -o pv_name,pv_size
  PV         PSize
  /dev/sdb1  17.14G
  /dev/sdc1  17.14G
  /dev/sdd1  17.14G
```

- Mit dem Pluszeichen (+) können Sie ein Feld zur Ausgabe hinzufügen. Dies wird in Kombination mit dem Parameter "-o" verwendet.

  Das folgende Beispiel zeigt die UUID des physischen Datenträgers zusätzlich zu den Standardfeldern an.

```
# pvs -o +pv_uuid
  PV         VG     Fmt  Attr PSize  PFree  PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-
dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G Joqlch-yWSj-kuEn-IdwM-01S9-X08M-
mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-
UqkCS
```

- Das Hinzufügen des Parameters **-v** zu einem Befehl umfasst einige zusätzliche Felder. Der Befehl **pvs -v** zeigt beispielsweise die Felder **DevSize** und **PV UUID** zusätzlich zu den Standardfeldern an.

```
# pvs -v
    Scanning for physical volume names
  PV         VG     Fmt  Attr PSize  PFree  DevSize PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G  17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G  17.14G Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G  17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

- Der Parameter **--noheadings** unterdrückt die Kopfzeile. Dies kann beim Schreiben von Skripten nützlich sein.

  Im folgenden Beispiel wird der Parameter **--noheadings** in Kombination mit dem Parameter **pv_name** verwendet und so eine Liste aller physischen Datenträger generiert.

```
# pvs --noheadings -o pv_name
  /dev/sdb1
  /dev/sdc1
  /dev/sdd1
```

- Der Parameter **--separator *separator*** verwendet ***separator***, um jedes Feld zu trennen.

  Im folgenden Beispiel werden die standardmäßigen Ausgabefelder des Befehls **pvs** mit einem Gleichheitszeichen (=) getrennt.

```
# pvs --separator =
  PV=VG=Fmt=Attr=PSize=PFree
  /dev/sdb1=new_vg=lvm2=a-=17.14G=17.14G
  /dev/sdc1=new_vg=lvm2=a-=17.14G=17.09G
  /dev/sdd1=new_vg=lvm2=a-=17.14G=17.14G
```

  Um die Felder bei der Verwendung des Parameters **separator** auszurichten, verwenden Sie den Parameter **separator** in Verbindung mit dem Parameter **--aligned**.

```
# pvs --separator = --aligned
  PV        =VG     =Fmt =Attr=PSize =PFree
  /dev/sdb1 =new_vg=lvm2=a-  =17.14G=17.14G
  /dev/sdc1 =new_vg=lvm2=a-  =17.14G=17.09G
  /dev/sdd1 =new_vg=lvm2=a-  =17.14G=17.14G
```

Sie können den Parameter **-P** der Befehle **lvs** oder **vgs** verwenden, um Informationen über einen ausgefallenen Datenträger anzuzeigen, der andernfalls nicht in der Ausgabe erscheinen würde. Werfen Sie einen Blick auf Abschnitt 6.2, „Anzeigen von Informationen auf ausgefallenen Geräten“ für Informationen zu der Ausgabe, die dieser Parameter erstellt.

Werfen Sie einen Blick auf die Handbuchseiten (8) der Befehle **pvs**, **vgs** und **lvs** für eine komplette Liste der Anzeigeparameter.

Felder von Datenträgergruppen können entweder mit Feldern von physischen Datenträgern (und physischen Datenträgersegmenten) oder mit Feldern von logischen Datenträgern (und logischen Datenträgersegmenten) gemischt werden. Felder von physischen und logischen Datenträgern können jedoch nicht gemischt werden. Der folgende Befehl zeigt beispielsweise eine Ausgabezeile für jeden

physischen Datenträger an.

```
# vgs -o +pv_name
  VG     #PV #LV #SN Attr   VSize  VFree  PV
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdc1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdd1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdb1
```

### 4.8.2. Objektauswahl

Dieser Abschnitt liefert eine Reihe an Tabellen, die die Informationen, die Sie über die LVM-Objekte mit den Befehlen **pvs**, **vgs** und **lvs** anzeigen können, auflistet.

Zur Vereinfachung kann ein Präfix für einen Feldnamen weggelassen werden, falls es dem Standard für den Befehl entspricht. In Zusammenhang mit dem Befehl **pvs** bedeutet **name** beispielsweise **pv_name**, jedoch wird beim Befehl **vgs** **name** als **vg_name** interpretiert.

Das Ausführen des folgenden Befehls entspricht dem Ausführen von **pvs -o pv_free**.

```
# pvs -o +free
  PFree
  17.14G
  17.09G
  17.14G
```

### Der pvs-Befehl

Tabelle 4.1, „pvs-Anzeigefelder“ listet die Anzeigeparameter des Befehls **pvs** zusammen mit dem Feldnamen (wie er in der Kopfanzeige erscheint) und einer Beschreibung des Feldes auf.

**Tabelle 4.1. pvs-Anzeigefelder**

| Parameter | Kopfzeile | Beschreibung |
|---|---|---|
| **dev_size** | DevSize | Die Größe des zugrunde liegenden Geräts, auf dem der physische Datenträger erstellt wurde |
| **pe_start** | 1st PE | Abstand bis zum Anfang des ersten Extents auf dem zugrunde liegenden Gerät |
| **pv_attr** | Attr | Status des physischen Datenträgers: (a)llocatable oder e(x)ported |
| **pv_fmt** | Fmt | Das Format der Metadaten des physischen Datenträgers (**lvm2** oder **lvm1**) |
| **pv_free** | PFree | Der verbleibende freie Platz auf dem physischen Datenträger |
| **pv_name** | PV | Der Name des physischen Datenträgers |
| **pv_pe_alloc_count** | Alloc | Anzahl der verwendeten physischen Extents |
| **pv_pe_count** | PE | Anzahl der physischen Extents |
| **pvseg_size** | SSize | Die Segmentgröße des physischen Datenträgers |
| **pvseg_start** | Start | Das physische Anfangs-Extent des physischen Datenträgersegments |
| **pv_size** | PSize | Die Größe des physischen Datenträgers |
| **pv_tags** | PV Tags | An den physischen Datenträger angehängte LVM-Tags |
| **pv_used** | Used | Die Menge an Platz, die derzeit auf dem physischen Datenträger verwendet wird |
| **pv_uuid** | PV UUID | Die UUID des physischen Datenträgers |

Der Befehl **pvs** zeigt standardmäßig die folgenden Felder an: **pv_name**, **vg_name**, **pv_fmt**, **pv_attr**, **pv_size**, **pv_free**. Die Anzeige wird nach **pv_name** sortiert.

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G
```

Durch die Verwendung des Parameters **-v** zusammen mit dem Befehl **pvs** werden die folgenden Felder zu der Standardanzeige hinzugefügt: **dev_size**, **pv_uuid**.

```
# pvs -v
    Scanning for physical volume names
  PV         VG     Fmt  Attr PSize  PFree  DevSize PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G  17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-
6XqA-dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G  17.14G Joqlch-yWSj-kuEn-IdwM-01S9-
XO8M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G  17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-
0dGW-tUqkCS
```

Mithilfe des Parameters **--segments** des Befehls **pvs** können Sie Informationen über jedes physische Datenträgersegment anzeigen. Ein Segment ist eine Gruppe von Extents. Eine Segmentanzeige kann nützlich sein, wenn Sie sehen möchten, ob Ihr logischer Datenträger fragmentiert ist.

Der Befehl **pvs --segments** zeigt standardmäßig die folgenden Felder an: **pv_name**, **vg_name**, **pv_fmt**, **pv_attr**, **pv_size**, **pv_free**, **pvseg_start**, **pvseg_size**. Die Anzeige wird innerhalb des physischen Datenträgers nach **pv_name** und **pvseg_size** sortiert.

```
# pvs --segments
  PV         VG         Fmt  Attr PSize  PFree  Start SSize
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M     0  1172
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1172    16
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1188     1
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G     0    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    26    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    50    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    76    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   100    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   126    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   150    22
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   172  4217
  /dev/sdb1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdc1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdd1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sde1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdf1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdg1  vg         lvm2 a-   17.14G 17.14G     0  4389
```

Mithilfe des Befehls **pvs -a** können Sie nachprüfen, welche Geräte, die von LVM erkannt wurden, nicht als physische LVM-Datenträger initialisiert wurden.

```
# pvs -a
  PV                             VG     Fmt  Attr PSize  PFree
  /dev/VolGroup00/LogVol01                   --       0      0
  /dev/new_vg/lvol0                          --       0      0
  /dev/ram                                   --       0      0
  /dev/ram0                                  --       0      0
  /dev/ram2                                  --       0      0
  /dev/ram3                                  --       0      0
  /dev/ram4                                  --       0      0
  /dev/ram5                                  --       0      0
  /dev/ram6                                  --       0      0
  /dev/root                                  --       0      0
  /dev/sda                                   --       0      0
  /dev/sdb                                   --       0      0
  /dev/sdb1                      new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc                                   --       0      0
  /dev/sdc1                      new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd                                   --       0      0
  /dev/sdd1                      new_vg lvm2 a-   17.14G 17.14G
```

## Der vgs-Befehl

Tabelle 4.2, „vgs-Anzeigefelder“ listet die Anzeigeparameter des Befehls **vgs** zusammen mit dem Feldnamen (wie er in der Kopfanzeige erscheint) und einer Beschreibung des Feldes auf.

**Tabelle 4.2. vgs-Anzeigefelder**

| Parameter | Kopfzeile | Beschreibung |
| --- | --- | --- |
| **lv_count** | #LV | Die Anzahl der logischen Datenträger, die die Datenträgergruppe enthält |
| **max_lv** | MaxLV | Die maximale Anzahl logischer Datenträger, die in der Datenträgergruppe erlaubt sind (0 falls unbegrenzt) |
| **max_pv** | MaxPV | Die maximale Anzahl physischer Datenträger, die in der Datenträgergruppe erlaubt sind (0 falls unbegrenzt) |
| **pv_count** | #PV | Die Anzahl der physischen Datenträger, die die Datenträgergruppe definieren |
| **snap_count** | #SN | Die Anzahl der Snapshots, die die Datenträgergruppe enthält |
| **vg_attr** | Attr | Status der Datenträgergruppe: (w)riteable, (r)eadonly, resi(z)eable, e(x)ported, (p)artial und (c)lustered. |
| **vg_extent_count** | #Ext | Die Anzahl der physischen Extents in der Datenträgergruppe |
| **vg_extent_size** | Ext | Die Größe der physischen Extents in der Datenträgergruppe |
| **vg_fmt** | Fmt | Das Format der Metadaten der Datenträgergruppe (**lvm2** or **lvm1**) |
| **vg_free** | VFree | Die Größe des verbleibenden freien Platzes in der Datenträgergruppe |
| **vg_free_count** | Free | Anzahl der freien physischen Extents in der Datenträgergruppe |
| **vg_name** | VG | Der Name der Datenträgergruppe |
| **vg_seqno** | Seq | Die Revisionsnummer der Datenträgergruppe |
| **vg_size** | VSize | Die Größe der Datenträgergruppe |
| **vg_sysid** | SYS ID | LVM1 System-ID |
| **vg_tags** | VG Tags | LVM-Tags, die an die Datenträgergruppe angefügt sind |
| **vg_uuid** | VG UUID | Die UUID der Datenträgergruppe |

Der Befehl **vgs** zeigt standardmäßig die folgenden Felder an: **vg_name**, **pv_count**, **lv_count**, **snap_count**, **vg_attr**, **vg_size**, **vg_free**. Die Anzeige wird nach **vg_name** sortiert.

```
# vgs
  VG     #PV #LV #SN Attr   VSize  VFree
  new_vg   3   1   1 wz--n- 51.42G 51.36G
```

Durch den Parameter **-v** zusammen mit dem Befehl **vgs** werden die folgenden Felder zur Standardanzeige hinzugefügt: **vg_extent_size**, **vg_uuid**.

```
# vgs -v
    Finding all volume groups
    Finding volume group "new_vg"
  VG     Attr   Ext   #PV #LV #SN VSize  VFree  VG UUID
  new_vg wz--n- 4.00M   3   1   1 51.42G 51.36G jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-
fD5D32
```

### Der lvs-Befehl

Tabelle 4.3, „lvs-Anzeigefelder“ listet die Anzeigeparameter des Befehls **lvs** zusammen mit dem Feldnamen (wie er in der Kopfanzeige erscheint) und einer Beschreibung des Feldes auf.

**Tabelle 4.3. lvs-Anzeigefelder**

| Parameter | Kopfzeile | Beschreibung |
|---|---|---|
| **chunksize**<br><br>**chunk_size** | Chunk | Größe einer Einheit in einem Snapshot-Datenträger |
| **copy_percent** | Copy% | Der Prozentsatz der Synchronisation eines gespiegelten logischen Datenträgers. Dies wird auch verwendet, wenn physische Extents mit dem Befehl **pv_move** verschoben werden |
| **devices** | Devices | Die zugrunde liegenden Geräte, aus dem der logische Datenträger besteht: die physischen und logischen Datenträger und der Anfang der physischen und logischen Extents |
| **lv_attr** | Attr | Der Status des logischen Datenträgers. Die Attribut-Bits des logischen Datenträgers lauten wie folgt:<br><br>Bit 1: Datenträgertyp: (m)irrored, (M)irrored without initial sync, (o)rigin, (p)vmove, (s)napshot, invalid (S)napshot, (v)irtual<br><br>Bit 2: Zugriffsrechte: (w)riteable, (r)ead-only<br><br>Bit 3: Zuweisungsrichtlinie: (c)ontiguous, (n)ormal, (a)nywhere, (i)nherited. Diese wird aktiviert, wenn der Datenträger derzeit für Zuweisungsänderungen gesperrt ist, wenn beispielsweise der Befehl **pvmove** ausgeführt wird.<br><br>Bit 4: festgesetzte (m)inor<br><br>Bit 5: Status: (a)ctive, (s)uspended, (I)nvalid snapshot, invalid (S)uspended snapshot, mapped (d)evice present without tables, mapped device present with (i)nactive table<br><br>Bit 6: Gerät (o)pen |
| **lv_kernel_major** | KMaj | Tatsächliche Major-Gerätenummer des logischen Datenträgers (-1 falls inaktiv) |
| **lv_kernel_minor** | KMIN | Tatsächliche Minor-Gerätenummer des logischen Datenträgers (-1 falls inaktiv) |
| **lv_major** | Maj | Die persistente Major-Gerätenummer des logischen Datenträgers (-1, falls nicht angegeben) |
| **lv_minor** | Min | Die persistente Minor-Gerätenummer des logischen Datenträgers (-1, falls nicht angegeben) |
| **lv_name** | LV | Der Name des logischen Datenträgers |
| **lv_size** | LSize | Die Größe des logischen Datenträgers |
| **lv_tags** | LV Tags | An den logischen Datenträger angehängte LVM-Tags |
| **lv_uuid** | LV UUID | Die UUID des logischen Datenträgers |
| **mirror_log** | Log | Gerät, auf dem sich die gespiegelte Protokolldatei |

| | | |
|---|---|---|
| | | befindet |
| **modules** | Modules | Entsprechendes Kernel Device-Mapper-Ziel, das für die Verwendung dieses logischen Datenträgers benötigt wird |
| **move_pv** | Move | Physischer Quell-Datenträger eines temporären logischen Datenträgers, der mit dem Befehl **pvmove** erstellt wurde |
| **origin** | Origin | Das ursprüngliche Gerät eines Snapshot-Datenträgers |
| **regionsize**<br><br>**region_size** | Region | Die Größe des Elements eines gespiegelten logischen Datenträgers |
| **seg_count** | #Seg | Die Anzahl der Segmente im logischen Datenträger |
| **seg_size** | SSize | Die Größe der Segmente im logischen Datenträger |
| **seg_start** | Start | Beginn des Segments auf dem logischen Datenträger |
| **seg_tags** | Seg Tags | An die Segmente des logischen Datenträger angehängte LVM-Tags |
| **segtype** | Typ | Der Segmenttyp eines logischen Datenträgers (zum Beispiel: mirror, striped, linear) |
| **snap_percent** | Snap% | Aktueller Prozentsatz eines Snapshot-Datenträgers, der verwendet wird |
| **stripes** | #Str | Anzahl der Stripes oder Mirrors auf einem logischen Datenträger |
| **stripesize**<br><br>**stripe_size** | Stripe | Größe einer Einheit des Stripes auf einem logischen Striped-Datenträger |

Der Befehl **lvs** zeigt standardmäßig die folgenden Felder an: **lv_name**, **vg_name**, **lv_attr**, **lv_size**, **origin**, **snap_percent**, **move_pv**, **mirror_log**, **copy_percent**. Die Standardanzeige wird innerhalb der Datenträgergruppe nach **vg_name** und **lv_name** sortiert.

```
# lvs
  LV         VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0      new_vg owi-a- 52.00M
  newvgsnap1 new_vg swi-a-  8.00M lvol0    0.20
```

Durch die Verwendung des Parameters **-v** zusammen mit dem Befehl **lvs** werden die folgenden Felder zu der Standardanzeige hinzugefügt: **seg_count**, **lv_major**, **lv_minor**, **lv_kernel_major**, **lv_kernel_minor**, **lv_uuid**.

```
# lvs -v
    Finding all logical volumes
  LV         VG     #Seg Attr   LSize  Maj Min KMaj KMin Origin Snap%  Move Copy%
Log LV UUID
  lvol0      new_vg    1 owi-a- 52.00M  -1  -1 253  3
LBy1Tz-sr23-OjsI-LT03-nHLC-y8XW-EhCl78
  newvgsnap1 new_vg    1 swi-a-  8.00M  -1  -1 253  5    lvol0    0.20
1ye1OU-1cIu-o79k-20h2-ZGF0-qCJm-CfbsIx
```

Mithilfe des Parameters **--segments** des Befehls **lvs** können Sie Informationen mit standardmäßigen

Spalten anzeigen, die die Segmentinformationen hervorheben. Bei der Verwendung des Parameters **`segments`** ist das Präfix **`seg`** optional. Der Befehl **`lvs --segments`** zeigt standardmäßig die folgenden Felder an: **`lv_name`**, **`vg_name`**, **`lv_attr`**, **`stripes`**, **`segtype`**, **`seg_size`**. Die Standardanzeige wird innerhalb der Datenträgergruppe nach **`vg_name`**, **`lv_name`** und innerhalb des logischen Datenträgers nach **`seg_start`** sortiert. Sind die logischen Datenträger fragmentiert, würde die Ausgabe dieses Befehls dies wiedergeben.

```
# lvs --segments
  LV       VG         Attr   #Str Type   SSize
  LogVol00 VolGroup00 -wi-ao    1 linear  36.62G
  LogVol01 VolGroup00 -wi-ao    1 linear 512.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear  88.00M
```

Wird der Parameter **`-v`** mit dem Befehl **`lvs --segments`** verwendet, werden die folgenden Felder zu der Standardanzeige hinzugefügt: **`seg_start`**, **`stripesize`**, **`chunksize`**.

```
# lvs -v --segments
    Finding all logical volumes
  LV         VG      Attr   Start SSize  #Str Type   Stripe Chunk
  lvol0      new_vg owi-a-     0  52.00M    1 linear     0     0
  newvgsnap1 new_vg swi-a-     0   8.00M    1 linear     0  8.00K
```

Das folgende Beispiel zeigt die Standardausgabe des Befehls **`lvs`** auf einem System mit einem logischen Datenträger, gefolgt von der Ausgabe des Befehls **`lvs`** zusammen mit dem Parameter **`segments`**.

```
# lvs
  LV    VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0 new_vg -wi-a- 52.00M
# lvs --segments
  LV    VG     Attr   #Str Type   SSize
  lvol0 new_vg -wi-a-    1 linear 52.00M
```

### 4.8.3. LVM-Berichte sortieren

Normalerweise muss die gesamte Ausgabe der Befehle **`lvs`**, **`vgs`** oder **`pvs`** intern generiert und gespeichert werden, bevor sie sortiert und Spalten korrekt ausgerichtet werden können. Mithilfe des Parameters **`--unbuffered`** können Sie die unsortierte Ausgabe im Moment der Generierung bereits anzeigen.

Verwenden Sie den Parameter **`-O`** von einem beliebigen Befehl zum Erstellen von Berichten, um eine alternative Anordnungsliste der zu sortierenden Spalten anzugeben. Es ist nicht notwendig, diese Felder innerhalb der Ausgabe selbst einzubinden.

Das folgende Beispiel zeigt die Ausgabe des Befehls **`pvs`**, die den Namen, die Größe und den freien Platz des physischen Datenträgers anzeigt.

```
# pvs -o pv_name,pv_size,pv_free
  PV         PSize  PFree
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
```

Das folgende Beispiel zeigt dieselbe Ausgabe, sortiert nach dem Feld für den freien Platz.

```
# pvs -o pv_name,pv_size,pv_free -O pv_free
  PV         PSize  PFree
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
```

Das folgende Beispiel zeigt, dass Sie das Feld, nach dem Sie sortieren, nicht anzeigen müssen.

```
# pvs -o pv_name,pv_size -O pv_free
  PV         PSize
  /dev/sdc1  17.14G
  /dev/sdd1  17.14G
  /dev/sdb1  17.14G
```

Um eine umgekehrte Sortierung anzuzeigen, fügen Sie ein **-** Zeichen vor dem Feld ein, das Sie nach dem Parameter **-O** eingeben.

```
# pvs -o pv_name,pv_size,pv_free -O -pv_free
  PV         PSize  PFree
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
```

### 4.8.4. Einheiten angeben

Verwenden Sie den Parameter **--units** des Befehls zum Erstellen des Berichts, um die Einheit für die Anzeige des LVM-Berichts anzugeben. Sie können (b)ytes, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (e)xabytes, (p)etabytes und (h)uman-readable angeben. Die Standardanzeige ist "human-readable", also einfach lesbar. Sie können den Standardwert übergehen, indem Sie den Parameter **units** im Abschnitt **global** der Datei **lvm.conf** setzen.

Das folgende Beispiel gibt die Ausgabe des Befehls **pvs** in Megabytes anstatt in Gigabytes (Standard) an.

```
# pvs --units m
  PV         VG     Fmt  Attr PSize     PFree
  /dev/sda1         lvm2 --    17555.40M 17555.40M
  /dev/sdb1  new_vg lvm2 a-   17552.00M 17552.00M
  /dev/sdc1  new_vg lvm2 a-   17552.00M 17500.00M
  /dev/sdd1  new_vg lvm2 a-   17552.00M 17552.00M
```

Standardmäßig werden Einheiten in Zweierpotenzen (Vielfaches von 1024) angezeigt. Sie können festlegen, dass Einheiten in einem Vielfachen von 1000 angezeigt werden, indem Sie die Spezifikationen für die Einheiten groß schreiben (B, K, M, G, T, H).

Der folgende Befehl zeigt die Ausgabe als ein Vielfaches von 1024 an (Standardverhalten).

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G
```

Der folgende Befehl zeigt die Ausgabe als ein Vielfaches von 1000.

```
#  pvs --units G
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   18.40G 18.40G
  /dev/sdc1  new_vg lvm2 a-   18.40G 18.35G
  /dev/sdd1  new_vg lvm2 a-   18.40G 18.40G
```

Sie können auch (s)ectors (Sektoren, definiert als 512 Bytes) oder angepasste Einheiten angeben.

Das folgende Beispiel zeigt die Ausgabe des Befehls **pvs** als eine Anzahl von Sektoren an.

```
# pvs --units s
  PV         VG     Fmt  Attr PSize     PFree
  /dev/sdb1  new_vg lvm2 a-   35946496S 35946496S
  /dev/sdc1  new_vg lvm2 a-   35946496S 35840000S
  /dev/sdd1  new_vg lvm2 a-   35946496S 35946496S
```

Der folgende Befehl zeigt die Ausgabe des Befehls **pvs** in Einheiten von 4 Megabytes an.

```
# pvs --units 4m
  PV         VG     Fmt  Attr PSize    PFree
  /dev/sdb1  new_vg lvm2 a-   4388.00U 4388.00U
  /dev/sdc1  new_vg lvm2 a-   4388.00U 4375.00U
  /dev/sdd1  new_vg lvm2 a-   4388.00U 4388.00U
```

# Kapitel 5. Konfigurationsbeispiele für LVM

Dieses Kapitel liefert einige grundlegende Konfigurationsbeispiele für LVM.

## 5.1. Erstellen eines logischen LVM-Datenträgers auf drei Platten

Dieses Beispiel erstellt einen logischen LVM-Datenträger mit der Bezeichnung **new_logical_volume**, der aus den Platten unter **/dev/sda1**, **/dev/sdb1** und **/dev/sdc1** besteht.

### 5.1.1. Erstellen der physischen Datenträger

Um Platten in einer Datenträgergruppe zu verwenden, kennzeichnen Sie diese als physische LVM-Datenträger.



**Warnung**

Dieser Befehl zerstört sämtliche Daten auf **/dev/sda1**, **/dev/sdb1** und **/dev/sdc1**.

```
[root@tng3-1 ~]# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdc1" successfully created
```

### 5.1.2. Erstellen der Datenträgergruppe

Der folgende Befehl erstellt die Datenträgergruppe **new_vol_group**.

```
[root@tng3-1 ~]# vgcreate new_vol_group /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "new_vol_group" successfully created
```

Sie können den Befehl **vgs** verwenden, um die Parameter der neuen Datenträgergruppe anzuzeigen.

```
[root@tng3-1 ~]# vgs
  VG            #PV #LV #SN Attr   VSize  VFree
  new_vol_group   3   0   0 wz--n- 51.45G 51.45G
```

### 5.1.3. Erstellen des logischen Datenträgers

Der folgende Befehl erstellt den logischen Datenträger **new_logical_volume** aus der Datenträgergruppe **new_vol_group**. Im Rahmen des folgenden Beispiels wird ein logischer Datenträger erstellt, der 2 GB der Datenträgergruppe verwendet.

```
[root@tng3-1 ~]# lvcreate -L2G -n new_logical_volume new_vol_group
  Logical volume "new_logical_volume" created
```

### 5.1.4. Erstellen des Dateisystems

Der folgende Befehl erstellt ein GFS2-Dateisystem auf dem logischen Datenträger.

```
[root@tng3-1 ~]# mkfs.gfs2 -plock_nolock -j 1
/dev/new_vol_group/new_logical_volume
This will destroy any data on /dev/new_vol_group/new_logical_volume.

Are you sure you want to proceed? [y/n] y

Device:                    /dev/new_vol_group/new_logical_volume
Blocksize:                 4096
Filesystem Size:           491460
Journals:                  1
Resource Groups:           8
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
All Done
```

Die folgenden Befehle hängen den logischen Datenträger ein und geben die Belegung des Plattenplatzes des Dateisystems wieder.

```
[root@tng3-1 ~]# mount /dev/new_vol_group/new_logical_volume /mnt
[root@tng3-1 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/new_vol_group/new_logical_volume
                       1965840        20   1965820   1% /mnt
```

## 5.2. Erstellen eines logischen Striped-Datenträgers

Dieses Beispiel erstellt einen logischen Striped-LVM-Datenträger mit der Bezeichnung **striped_logical_volume**, der Daten zwischen den Platten unter **/dev/sda1**, **/dev/sdb1** und **/dev/sdc1** verteilt.

### 5.2.1. Erstellen der physischen Datenträger

Kennzeichnen Sie die Platten, die Sie in den Datenträgergruppen verwenden werden, als physische LVM-Datenträger.



**Warnung**

Dieser Befehl zerstört sämtliche Daten auf **/dev/sda1**, **/dev/sdb1** und **/dev/sdc1**.

```
[root@tng3-1 ~]# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdc1" successfully created
```

### 5.2.2. Erstellen der Datenträgergruppe

Der folgende Befehl erstellt die Datenträgergruppe **volgroup01**.

```
[root@tng3-1 ~]# vgcreate volgroup01 /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "volgroup01" successfully created
```

Sie können den Befehl **vgs** verwenden, um die Parameter der neuen Datenträgergruppe anzuzeigen.

```
[root@tng3-1 ~]# vgs
  VG                #PV #LV #SN Attr   VSize  VFree
  volgroup01          3   0   0 wz--n- 51.45G 51.45G
```

### 5.2.3. Erstellen des logischen Datenträgers

Der folgende Befehl erstellt den logischen Striped-Datenträger **striped_logical_volume** aus der Datenträgergruppe **volgroup01**. Im Rahmen des folgenden Beispiels wird ein logischer Datenträger mit einer Größe von 2 GB und mit drei Stripes und einer Größe von 4 Kilobytes pro Stripe erstellt.

```
[root@tng3-1 ~]# lvcreate -i3 -I4 -L2G -nstriped_logical_volume volgroup01
  Rounding size (512 extents) up to stripe boundary size (513 extents)
  Logical volume "striped_logical_volume" created
```

### 5.2.4. Erstellen des Dateisystems

Der folgende Befehl erstellt ein GFS2-Dateisystem auf dem logischen Datenträger.

```
[root@tng3-1 ~]# mkfs.gfs2 -plock_nolock -j 1
/dev/volgroup01/striped_logical_volume
This will destroy any data on /dev/volgroup01/striped_logical_volume.

Are you sure you want to proceed? [y/n] y

Device:                    /dev/volgroup01/striped_logical_volume
Blocksize:                 4096
Filesystem Size:           492484
Journals:                  1
Resource Groups:           8
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
All Done
```

Die folgenden Befehle hängen den logischen Datenträger ein und geben die Belegung des Plattenplatzes des Dateisystems wieder.

```
[root@tng3-1 ~]# mount /dev/volgroup01/striped_logical_volume /mnt
[root@tng3-1 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      13902624   1656776  11528232  13% /
/dev/hda1               101086     10787     85080  12% /boot
tmpfs                   127880         0    127880   0% /dev/shm
/dev/volgroup01/striped_logical_volume
                       1969936        20   1969916   1% /mnt
```

## 5.3. Aufteilen einer Datenträgergruppe

In diesem Beispiel besteht eine existierende Datenträgergruppe aus drei physischen Datenträgern. Falls genug unbenutzter Platz auf den physischen Datenträgern vorhanden ist, kann eine neue Datenträgergruppe erstellt werden, ohne neue Platten hinzuzufügen.

Bei der erstmaligen Einrichtung wird der logische Datenträger **mylv** aus der Datenträgergruppe **myvol** geschaffen, welche wiederum aus den drei physischen Datenträgern **/dev/sda1**, **/dev/sdb1** und **/dev/sdc1** besteht.

Nach Abschluss dieser Prozedur besteht die Datenträgergruppe **myvg** aus **/dev/sda1** und **/dev/sdb1**. Eine zweite Datenträgergruppe **yourvg** besteht aus **/dev/sdc1**.

### 5.3.1. Ermitteln von freiem Speicherplatz

Mithilfe des Befehls **pvscan** können Sie ermitteln, wie viel freier Speicherplatz derzeit in der Datenträgergruppe zur Verfügung steht.

```
[root@tng3-1 ~]# pvscan
  PV /dev/sda1  VG myvg   lvm2 [17.15 GB / 0    free]
  PV /dev/sdb1  VG myvg   lvm2 [17.15 GB / 12.15 GB free]
  PV /dev/sdc1  VG myvg   lvm2 [17.15 GB / 15.80 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
```

### 5.3.2. Verschieben der Daten

Sie können alle verwendeten physischen Extents in **/dev/sdc1** mit dem Befehl **pvmove** nach **/dev/sdb1** verschieben. **pvmove** kann bei der Ausführung einige Zeit in Anspruch nehmen.

```
[root@tng3-1 ~]# pvmove /dev/sdc1 /dev/sdb1
  /dev/sdc1: Moved: 14.7%
  /dev/sdc1: Moved: 30.3%
  /dev/sdc1: Moved: 45.7%
  /dev/sdc1: Moved: 61.0%
  /dev/sdc1: Moved: 76.6%
  /dev/sdc1: Moved: 92.2%
  /dev/sdc1: Moved: 100.0%
```

Nach dem Verschieben der Daten sehen Sie, dass sämtlicher Platz auf **/dev/sdc1** frei ist.

```
[root@tng3-1 ~]# pvscan
  PV /dev/sda1   VG myvg   lvm2 [17.15 GB / 0    free]
  PV /dev/sdb1   VG myvg   lvm2 [17.15 GB / 10.80 GB free]
  PV /dev/sdc1   VG myvg   lvm2 [17.15 GB / 17.15 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
```

### 5.3.3. Aufteilen der Datenträgergruppe

Um eine neue Datenträgergruppe **yourvg** zu erstellen, verwenden Sie zur Aufteilung der Datenträgergruppe **myvg** den Befehl **vgsplit**.

Bevor Sie die Datenträgergruppe aufteilen können, muss der logische Datenträger inaktiv sein. Falls das Dateisystem eingehängt ist, müssen Sie es vor der Deaktivierung des logischen Datenträgers aushängen.

Mithilfe der Befehle **lvchange** oder **vgchange** können Sie den logischen Datenträger deaktivieren. Der folgende Befehl deaktiviert den logischen Datenträger **mylv** und trennt die Datenträgergruppe **yourvg** von der Datenträgergruppe **myvg**, indem es den physischen Datenträger **/dev/sdc1** in die neue Datenträgergruppe **yourvg** verschiebt.

```
[root@tng3-1 ~]# lvchange -a n /dev/myvg/mylv
[root@tng3-1 ~]# vgsplit myvg yourvg /dev/sdc1
  Volume group "yourvg" successfully split from "myvg"
```

Mithilfe des Befehls **vgs** können Sie die Parameter der beiden Datenträgergruppen ansehen.

```
[root@tng3-1 ~]# vgs
  VG      #PV #LV #SN Attr   VSize  VFree
  myvg      2   1   0 wz--n- 34.30G 10.80G
  yourvg    1   0   0 wz--n- 17.15G 17.15G
```

### 5.3.4. Erstellen des neuen logischen Datenträgers

Nach dem Erstellen der neuen Datenträgergruppe können Sie den neuen logischen Datenträger **yourlv** erstellen.

```
[root@tng3-1 ~]# lvcreate -L5G -n yourlv yourvg
  Logical volume "yourlv" created
```

### 5.3.5. Erstellen eines Dateisystems und Einhängen des neuen logischen Datenträgers

Sie können ein Dateisystem auf dem neuen logischen Datenträger erstellen und es einhängen.

```
[root@tng3-1 ~]#  mkfs.gfs2 -plock_nolock -j 1 /dev/yourvg/yourlv
This will destroy any data on /dev/yourvg/yourlv.

Are you sure you want to proceed? [y/n] y

Device:                    /dev/yourvg/yourlv
Blocksize:                 4096
Filesystem Size:           1277816
Journals:                  1
Resource Groups:           20
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
All Done

[root@tng3-1 ~]# mount /dev/yourvg/yourlv /mnt
```

### 5.3.6. Aktivieren und Einhängen des originalen logischen Datenträgers

Da Sie den logischen Datenträger **mylv** deaktivieren mussten, müssen Sie ihn wieder aktivieren, bevor Sie ihn einhängen können.

```
root@tng3-1 ~]# lvchange -a y mylv

[root@tng3-1 ~]# mount /dev/myvg/mylv /mnt
[root@tng3-1 ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/yourvg/yourlv    24507776        32  24507744   1% /mnt
/dev/myvg/mylv        24507776        32  24507744   1% /mnt
```

## 5.4. Entfernen einer Platte aus einem logischen Datenträger

Dieses Beispiel zeigt, wie Sie eine Platte aus einem existierenden logischen Datenträger entfernen können, um entweder die Platte zu ersetzen, oder die Platte als Teil eines anderen Datenträgers zu verwenden. Um eine Platte zu entfernen, müssen Sie zunächst die Extents auf dem physischen LVM-Datenträger auf eine andere Platte oder eine Reihe von anderen Platten verschieben.

### 5.4.1. Verschieben von Extents auf existierende physische Datenträger

In diesem Beispiel wird der logische Datenträger auf vier physischen Datenträger in der Datenträgergruppe **myvg** verteilt.

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
```

Die Extents von **/dev/sdb1** sollen davon entfernt und woanders hin verschoben werden, so dass sie aus der Datenträgergruppe entfernt werden können.

Falls genügend freie Extents auf den anderen physischen Datenträgern in der Datenträgergruppe existieren, können Sie den Befehl **pvmove** ohne weitere Optionen auf dem Gerät ausführen, das Sie entfernen möchten, und die Extents werden auf die anderen Geräte verteilt.

```
[root@tng3-1 ~]# pvmove /dev/sdb1
  /dev/sdb1: Moved: 2.0%
 ...
  /dev/sdb1: Moved: 79.2%
 ...
  /dev/sdb1: Moved: 100.0%
```

Nach Abschluss der Ausführung von **pvmove** lautet die Verteilung der Extents wie folgt:

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 17.15G     0
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
```

Verwenden Sie den Befehl **vgreduce**, um den physischen Datenträger **/dev/sdb1** aus der Datenträgergruppe zu entfernen.

```
[root@tng3-1 ~]# vgreduce myvg /dev/sdb1
  Removed "/dev/sdb1" from volume group "myvg"
[root@tng3-1 ~]# pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G
  /dev/sdb1       lvm2 --   17.15G 17.15G
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G
```

Die Platte kann nun physisch entfernt oder anderen Benutzern zugewiesen werden.

### 5.4.2. Extents auf eine neue Platte verschieben

In diesem Beispiel wird der logische Datenträger unter den drei physischen Datenträgern in der Datenträgergruppe **myvg** wie folgt verteilt:

```
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 15.15G  2.00G
```

Die Extents von **/dev/sdb1** sollen auf ein neues Gerät **/dev/sdd1** verschoben werden.

#### 5.4.2.1. Erstellen des neuen physischen Datenträgers

Erstellen Sie einen neuen physischen Datenträger aus **/dev/sdd1**.

```
[root@tng3-1 ~]# pvcreate /dev/sdd1
  Physical volume "/dev/sdd1" successfully created
```

#### 5.4.2.2. Hinzufügen des neuen physischen Datenträgers zu der Datenträgergruppe

Fügen Sie **/dev/sdd1** zur vorhandenen Datenträgergruppe **myvg** hinzu.

```
[root@tng3-1 ~]# vgextend myvg /dev/sdd1
  Volume group "myvg" successfully extended
[root@tng3-1]# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1   myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdc1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdd1   myvg lvm2 a-   17.15G 17.15G     0
```

#### 5.4.2.3. Verschieben der Daten

Verwenden Sie **pvmove**, um Daten von **/dev/sdb1** auf **/dev/sdd1** zu verschieben.

```
[root@tng3-1 ~]# pvmove /dev/sdb1 /dev/sdd1
  /dev/sdb1: Moved: 10.0%
...
  /dev/sdb1: Moved: 79.7%
...
  /dev/sdb1: Moved: 100.0%

[root@tng3-1]# pvs -o+pv_used
  PV          VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1   myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1   myvg lvm2 a-   17.15G 17.15G     0
  /dev/sdc1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdd1   myvg lvm2 a-   17.15G 15.15G  2.00G
```

#### 5.4.2.4. Entfernen des alten physischen Datenträgers aus der Datenträgergruppe

Nachdem Sie die Daten von **/dev/sdb1** verschoben haben, können Sie dies aus der Datenträgergruppe entfernen.

```
[root@tng3-1 ~]# vgreduce myvg /dev/sdb1
  Removed "/dev/sdb1" from volume group "myvg"
```

Sie können nun die Platte einer anderen Datenträgergruppe zuweisen oder die Platte aus dem System entfernen.

## 5.5. Erstellen eines gespiegelten logischen LVM-Datenträgers in einem Cluster

Zum Erstellen eines gespiegelten logischen LVM-Datenträgers in einem Cluster verwenden Sie dieselben Befehle und Verfahren wie zum Erstellen eines gespiegelten logischen LVM-Datenträgers auf einem einzigen Knoten. Um jedoch einen gespiegelten logischen LVM-Datenträger in einem Cluster zu erzeugen, muss der Cluster und die Cluster-Spiegelinfrastruktur laufen, der Cluster muss einsatzbereit sein, und der Sperrtyp in der **lvm.conf**-Datei muss richtig eingestellt sein, um Cluster-Sperren zu aktivieren, entweder direkt oder mittels **lvmconf**-Befehl wie in Abschnitt 3.1, „LVM-Datenträger in einem Cluster erstellen“ beschrieben.

Das folgende Verfahren erstellt einen gespiegelten LVM-Datenträger in einem Cluster. Zunächst wird bei diesem Verfahren überprüft, ob die Cluster-Dienste installiert sind und ausgeführt werden, anschließend wird der gespiegelte Datenträger erstellt.

1. Um einen gespiegelten logischen Datenträger zu erstellen, der von allen Knoten in einem Cluster gemeinsam verwendet wird, muss der Sperrtyp in der **lvm.conf**-Datei für jeden Knoten korrekt eingestellt sein. Standardmäßig ist der Sperrtyp auf lokal gesetzt. Führen Sie auf jedem Knoten im Cluster folgenden Befehl aus, um dies zu ändern und geclusterte Sperren zu aktivieren:

   ```
   # /sbin/lvmconf --enable-cluster
   ```

2. Um einen geclusterten logischen Datenträger zu erstellen, muss die Cluster-Infrastruktur auf jedem Knoten im Cluster ausgeführt werden. Im folgenden Beispiel wird überprüft, ob der **clvmd**-Daemon auf dem Knoten läuft, auf dem er initiiert wurde.

   ```
   [root@doc-07 ~]# ps auxw | grep clvmd
   root     17642  0.0  0.1 32164 1072 ?        Ssl  Apr06   0:00 clvmd -T20 -t
   90
   ```

   Der folgende Befehl zeigt die lokale Ansicht des Cluster-Zustands:

   ```
   [root@example-01 ~]# cman_tool services
   fence domain
   member count  3
   victim count  0
   victim now    0
   master nodeid 2
   wait state    none
   members       1 2 3

   dlm lockspaces
   name          clvmd
   id            0x4104eefa
   flags         0x00000000
   change        member 3 joined 1 remove 0 failed 0 seq 1,1
   members       1 2 3
   ```

3. Vergewissern Sie sich, dass das **cmirror**-Paket installiert ist.
4. Starten Sie den **cmirrord**-Dienst.

```
[root@hexample-01 ~]# service cmirrord start
Starting cmirrord:                                          [  OK  ]
```

5. Erstellen Sie den Mirror. Erzeugen Sie dazu zunächst die physischen Datenträger. Die folgenden Befehle erzeugen drei physische Datenträger. Zwei der physischen Datenträger werden als Standbeine des Mirrors verwendet, der dritte physische Datenträger wird das Mirror-Protokoll enthalten.

```
[root@doc-07 ~]# pvcreate /dev/xvdb1
  Physical volume "/dev/xvdb1" successfully created
[root@doc-07 ~]# pvcreate /dev/xvdb2
  Physical volume "/dev/xvdb2" successfully created
[root@doc-07 ~]# pvcreate /dev/xvdc1
  Physical volume "/dev/xvdc1" successfully created
```

6. Erstellen Sie die Datenträgergruppe. Dieses Beispiel erzeugt eine Datenträgergruppe namens **vg001**, die aus den drei physischen Datenträgern besteht, die im vorangegangenen Schritt angelegt wurden.

```
[root@doc-07 ~]# vgcreate vg001 /dev/xvdb1 /dev/xvdb2 /dev/xvdc1
  Clustered volume group "vg001" successfully created
```

Beachten Sie, dass die Ausgabe des **vgcreate**-Befehls anzeigt, ob die Datenträgergruppe geclustert ist. Sie können überprüfen, ob eine Datenträgergruppe geclustert ist, indem Sie den **vgs**-Befehl ausführen, um die Parameter der Datenträgergruppe einzusehen. Ist die Datenträgergruppe geclustert, wird der "c"-Parameter angezeigt.

```
[root@doc-07 ~]# vgs vg001
  VG       #PV #LV #SN Attr   VSize  VFree
  vg001      3   0   0 wz--nc 68.97G 68.97G
```

7. Erstellen Sie den gespiegelten logischen Datenträger. Dieses Beispiel erzeugt den logischen Datenträger **mirrorlv** aus der Datenträgergruppe **vg001**. Dieser Datenträger hat ein Mirror-Standbein. Dieses Beispiel spezifiziert, welche Extents des physischen Datenträgers für den logischen Datenträger verwendet werden.

```
[root@doc-07 ~]# lvcreate -l 1000 -m1 vg001 -n mirrorlv /dev/xvdb1:1-1000
/dev/xvdb2:1-1000 /dev/xvdc1:0
  Logical volume "mirrorlv" created
```

Sie können den **lvs**-Befehl nutzen, um den Fortschritt der Mirror-Erstellung anzuzeigen. Das folgende Beispiel zeigt, dass der Mirror zu 47% synchronisiert ist, dann 91%, und schließlich zu 100% synchronisiert, wenn der Mirror vollständig ist.

```
[root@doc-07 log]# lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log          Copy%
Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog     47.00
[root@doc-07 log]# lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log          Copy%
Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog     91.00
[root@doc-07 ~]#  lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log          Copy%
Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog    100.00
```

Der Abschluss der Mirror-Erstellung wird im Systemprotokoll vermerkt:

```
May 10 14:52:52 doc-07 [19402]: Monitoring mirror device vg001-mirrorlv for
events
May 10 14:55:00 doc-07 lvm[19402]: vg001-mirrorlv is now in-sync
```

8. Sie können **lvs** zusammen mit den **-o +devices**-Optionen verwenden, um die Konfiguration des Mirrors anzuzeigen, z.B. welche Geräte die Mirror-Standbeine stellen. Wie Sie sehen, besteht der logische Datenträger in diesem Beispiel aus zwei linearen Images und einer Protokolldatei.

```
[root@doc-07 ~]# lvs -a -o +devices
  LV                   VG         Attr   LSize  Origin Snap%  Move Log
Copy%  Convert Devices
  mirrorlv             vg001      mwi-a-  3.91G
mirrorlv_mlog 100.00         mirrorlv_mimage_0(0),mirrorlv_mimage_1(0)
  [mirrorlv_mimage_0] vg001      iwi-ao  3.91G
/dev/xvdb1(1)
  [mirrorlv_mimage_1] vg001      iwi-ao  3.91G
/dev/xvdb2(1)
  [mirrorlv_mlog]     vg001      lwi-ao  4.00M
/dev/xvdc1(0)
```

Sie können die **seg_pe_ranges**-Option des **lvs**-Befehls verwenden, um das Daten-Layout anzuzeigen. Mithilfe dieser Option können Sie sich vergewissern, dass Ihr Layout einwandfrei redundant ist. Die Ausgabe dieses Befehls zeigt PE-Bereiche in demselben Format an, das die **lvcreate**- und **lvresize**-Befehle als Eingabe akzeptieren.

```
[root@doc-07 ~]# lvs -a -o +seg_pe_ranges --segments
  PE Ranges
  mirrorlv_mimage_0:0-999 mirrorlv_mimage_1:0-999
  /dev/xvdb1:1-1000
  /dev/xvdb2:1-1000
  /dev/xvdc1:0-0
```



## Anmerkung

Informationen über die Wiederherstellung nach dem Ausfall eines der Standbeine eines gespiegelten LVM-Datenträgers finden Sie in Abschnitt 6.3, „Wiederherstellung beim Ausfall eines LVM-Mirrors“.

# Kapitel 6. Suche und Bereinigung von LVM-Fehlern

Dieses Kapitel liefert Anweisungen zur Suche und Bereinigung von einer Vielzahl von LVM-Problemen.

## 6.1. Diagnostik zur Suche und Bereinigung von Fehlern

Falls ein Befehl nicht wie erwartet funktioniert, können Sie die Fehlersuche wie folgt durchführen:

- Verwenden Sie die Optionen **-v**, **-vv**, **-vvv** oder **-vvvv** von jedem beliebigen Befehl für mehr Details bei der Ausgabe.
- Falls das Problem mit der Aktivierung des logischen Datenträgers zusammenhängt, setzen Sie 'activation = 1' im Abschnitt 'log' der Konfigurationsdatei und führen den Befehl mit der Option **-vvvv** aus. Nachdem Sie mit der Untersuchung der Ausgabe fertig sind, stellen Sie sicher, diesen Parameter auf 0 zu setzen, um mögliche Probleme mit der Sperrung der Maschine während Situationen mit geringem Speicher zu vermeiden.
- Führen Sie den Befehl **lvmdump** aus, der eine Zusammenfassung von Informationen zu Diagnosezwecken liefert. Werfen Sie einen Blick auf die Handbuchseite (8) von **lvmdump** für weitere Informationen.
- Führen Sie die Befehle **lvs -v**, **pvs -a** oder **dmsetup info -c** für zusätzliche Systeminformationen aus.
- Untersuchen Sie die letzte Sicherung der Metadaten in **/etc/lvm/backup** und archivierte Versionen in **/etc/lvm/archive**.
- Überprüfen Sie die aktuellen Informationen zur Konfiguration, indem Sie **lvm dumpconfig** ausführen.
- Suchen Sie in der Datei **.cache** in **/etc/lvm** nach einem Eintrag, welche Geräte physische Datenträger aufweisen.

## 6.2. Anzeigen von Informationen auf ausgefallenen Geräten

Mithilfe der Option **-P** der Befehle **lvs** oder **vgs** können Sie Informationen zu einem ausgefallenen Datenträger, der andernfalls nicht in der Ausgabe erscheinen würde, anzeigen. Dieser Parameter ermöglicht einige Operationen, auch wenn die Metadaten intern nicht vollständig konsistent sind. Wenn beispielsweise eines der Geräte, aus dem die Datenträgergruppe **vg** besteht, ausfiele, könnte der Befehl **vgs** die folgende Ausgabe produzieren.

```
[root@link-07 tmp]# vgs -o +devices
  Volume group "vg" not found
```

Wenn Sie die Option **-P** des Befehls **vgs** angeben, kann die Datenträgergruppe zwar nach wie vor nicht verwendet werden, Sie erhalten jedoch mehr Informationen über das ausgefallene Gerät.

```
[root@link-07 tmp]# vgs -P -o +devices
  Partial mode. Incomplete volume groups will be activated read-only.
  VG   #PV #LV #SN Attr   VSize VFree Devices
  vg     9   2   0 rz-pn- 2.11T 2.07T unknown device(0)
  vg     9   2   0 rz-pn- 2.11T 2.07T unknown device(5120),/dev/sda1(0)
```

In diesem Beispiel führte der Ausfall des Geräts sowohl zu einem Ausfall eines linearen, als auch eines logischen Striped-Datenträgers in der Datenträgergruppe. Der Befehl **lvs** ohne die Option **-P** zeigt die folgende Ausgabe.

```
[root@link-07 tmp]# lvs -a -o +devices
  Volume group "vg" not found
```

Mithilfe der Option **-P** werden logische Datenträger angezeigt, die ausgefallen sind.

```
[root@link-07 tmp]# lvs -P -a -o +devices
  Partial mode. Incomplete volume groups will be activated read-only.
  LV     VG   Attr   LSize  Origin Snap%  Move Log Copy%  Devices
  linear vg   -wi-a- 20.00G                               unknown device(0)
  stripe vg   -wi-a- 20.00G                               unknown
device(5120),/dev/sda1(0)
```

Die folgenden Beispiele zeigen die Ausgabe der Befehle **pvs** und **lvs** mit der Option **-P** beim Ausfall eines Standbeins eines gespiegelten logischen Datenträgers.

```
root@link-08 ~]#  vgs -a -o +devices -P
  Partial mode. Incomplete volume groups will be activated read-only.
  VG    #PV #LV #SN Attr   VSize VFree Devices
  corey   4   4   0 rz-pnc 1.58T 1.34T
my_mirror_mimage_0(0),my_mirror_mimage_1(0)
  corey   4   4   0 rz-pnc 1.58T 1.34T /dev/sdd1(0)
  corey   4   4   0 rz-pnc 1.58T 1.34T unknown device(0)
  corey   4   4   0 rz-pnc 1.58T 1.34T /dev/sdb1(0)
```

```
[root@link-08 ~]# lvs -a -o +devices -P
  Partial mode. Incomplete volume groups will be activated read-only.
  LV                   VG    Attr   LSize   Origin Snap%  Move Log
Copy%  Devices
  my_mirror            corey mwi-a- 120.00G                my_mirror_mlog
1.95 my_mirror_mimage_0(0),my_mirror_mimage_1(0)
  [my_mirror_mimage_0] corey iwi-ao 120.00G
unknown device(0)
  [my_mirror_mimage_1] corey iwi-ao 120.00G
/dev/sdb1(0)
  [my_mirror_mlog]     corey lwi-ao   4.00M
/dev/sdd1(0)
```

## 6.3. Wiederherstellung beim Ausfall eines LVM-Mirrors

Dieser Abschnitt liefert ein Beispiel für die Wiederherstellung in einer Situation, in der ein Standbein eines gespiegelten LVM-Datenträgers ausfällt, da das zugrunde liegende Gerät für einen physischen Datenträger ausfällt und der **mirror_log_fault_policy**-Parameter auf **remove** gesetzt ist, weshalb Sie den Mirror manuell neu erstellen müssen. Für Informationen über das Einstellen des **mirror_log_fault_policy**-Parameters, siehe Abschnitt 6.3, „Wiederherstellung beim Ausfall eines LVM-Mirrors“.

Beim Ausfall eines Mirror-Standbeins konvertiert LVM den gespiegelten Datenträger in einen linearen Datenträger, der wie gewohnt weiterarbeitet, jedoch ohne die gespiegelte Redundanz. An dieser Stelle können Sie ein neues Plattengerät zum System hinzufügen, welches als Ersatz des physischen Geräts verwendet werden kann und anschließend den Mirror neu erstellen.

Der folgende Befehl erstellt den physischen Datenträger, der für den Mirror verwendet wird.

```
[root@link-08 ~]# pvcreate /dev/sd[abcdefgh][12]
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sda2" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdb2" successfully created
  Physical volume "/dev/sdc1" successfully created
  Physical volume "/dev/sdc2" successfully created
  Physical volume "/dev/sdd1" successfully created
  Physical volume "/dev/sdd2" successfully created
  Physical volume "/dev/sde1" successfully created
  Physical volume "/dev/sde2" successfully created
  Physical volume "/dev/sdf1" successfully created
  Physical volume "/dev/sdf2" successfully created
  Physical volume "/dev/sdg1" successfully created
  Physical volume "/dev/sdg2" successfully created
  Physical volume "/dev/sdh1" successfully created
  Physical volume "/dev/sdh2" successfully created
```

Die folgenden Befehle erstellen die Datenträgergruppe **vg** und den gespiegelten Datenträger **groupfs**.

```
[root@link-08 ~]# vgcreate vg /dev/sd[abcdefgh][12]
  Volume group "vg" successfully created
[root@link-08 ~]# lvcreate -L 750M -n groupfs -m 1 vg /dev/sda1 /dev/sdb1
/dev/sdc1
  Rounding up size to full physical extent 752.00 MB
  Logical volume "groupfs" created
```

Mithilfe des Befehls **lvs** können Sie das Layout des gespiegelten Datenträgers, sowie das zugrunde liegende Gerät für das Mirror-Standbein und die Protokolldatei des Mirrors verifizieren. Beachten Sie, dass der Mirror im ersten Beispiel noch nicht vollständig synchronisiert ist. Sie sollten abwarten, bis das Feld **Copy%** 100.00 anzeigt, bevor Sie fortfahren.

```
[root@link-08 ~]# lvs -a -o +devices
  LV                 VG   Attr   LSize   Origin Snap%  Move Log          Copy%
Devices
  groupfs            vg   mwi-a- 752.00M                    groupfs_mlog 21.28
groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sda1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
/dev/sdb1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M
/dev/sdc1(0)

[root@link-08 ~]# lvs -a -o +devices
  LV                 VG   Attr   LSize   Origin Snap%  Move Log          Copy%
Devices
  groupfs            vg   mwi-a- 752.00M                    groupfs_mlog 100.00
groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sda1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
/dev/sdb1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M     i
/dev/sdc1(0)
```

In diesem Beispiel fällt das primäre Standbein des Mirrors **/dev/sda1** aus. Jegliche Schreibaktivitäten

auf den gespiegelten Datenträger führen dazu, dass LVM den ausgefallenen Mirror entdeckt. Tritt dies auf, konvertiert LVM den Mirror in einen einzelnen, linearen Datenträger. Um in diesem Fall eine Konvertierung auszulösen, führen wir den Befehl **dd** aus.

```
[root@link-08 ~]# dd if=/dev/zero of=/dev/vg/groupfs count=10
10+0 records in
10+0 records out
```

Mithilfe des Befehls **lvs** können Sie verifizieren, dass es sich bei dem Gerät nun um ein lineares Gerät handelt. Aufgrund der ausgefallenen Platte treten I/O-Fehler auf.

```
[root@link-08 ~]# lvs -a -o +devices
  /dev/sda1: read failed after 0 of 2048 at 0: Input/output error
  /dev/sda2: read failed after 0 of 2048 at 0: Input/output error
  LV      VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  groupfs vg   -wi-a- 752.00M                               /dev/sdb1(0)
```

Zu diesem Zeitpunkt sollten Sie weiterhin in der Lage sein, den logischen Datenträger zu verwenden. Es ist jedoch keine Mirror-Redundanz vorhanden.

Um den gespiegelten Datenträger neu zu erstellen, ersetzen Sie das defekte Laufwerk und erstellen den physischen Datenträger erneut. Falls Sie dieselbe Platte verwenden, anstatt sie durch eine neue zu ersetzen, erhalten Sie beim Ausführen des Befehls **pvcreate** "inconsistent"-Warnungen. Sie können diese Warnungen vermeiden, indem Sie den Befehl **vgreduce --removemissing** ausführen.

```
[root@link-08 ~]# pvcreate /dev/sdi[12]
  Physical volume "/dev/sdi1" successfully created
  Physical volume "/dev/sdi2" successfully created

[root@link-08 ~]# pvscan
  PV /dev/sdb1   VG vg   lvm2 [67.83 GB / 67.10 GB free]
  PV /dev/sdb2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdi1           lvm2 [603.94 GB]
  PV /dev/sdi2           lvm2 [603.94 GB]
  Total: 16 [2.11 TB] / in use: 14 [949.65 GB] / in no VG: 2 [1.18 TB]
```

Als Nächstes können Sie die ursprüngliche Datenträgergruppe mit dem neuen physischen Datenträger erweitern.

```
[root@link-08 ~]# vgextend vg /dev/sdi[12]
  Volume group "vg" successfully extended

[root@link-08 ~]# pvscan
  PV /dev/sdb1   VG vg   lvm2 [67.83 GB / 67.10 GB free]
  PV /dev/sdb2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdc2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdd2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sde2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdf2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdg2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh1   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdh2   VG vg   lvm2 [67.83 GB / 67.83 GB free]
  PV /dev/sdi1   VG vg   lvm2 [603.93 GB / 603.93 GB free]
  PV /dev/sdi2   VG vg   lvm2 [603.93 GB / 603.93 GB free]
  Total: 16 [2.11 TB] / in use: 16 [2.11 TB] / in no VG: 0 [0   ]
```

Konvertieren Sie den linearen Datenträger zurück in seinen originalen, gespiegelten Zustand.

```
[root@link-08 ~]# lvconvert -m 1 /dev/vg/groupfs /dev/sdi1 /dev/sdb1 /dev/sdc1
  Logical volume mirror converted.
```

Mithilfe des Befehls **lvs** können Sie verifizieren, dass der Mirror wiederhergestellt wurde.

```
[root@link-08 ~]# lvs -a -o +devices
  LV                 VG   Attr   LSize   Origin Snap%  Move Log          Copy%
Devices
  groupfs            vg   mwi-a- 752.00M                    groupfs_mlog 68.62
groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sdb1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
/dev/sdi1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M
/dev/sdc1(0)
```

## 6.4. Wiederherstellen von Metadaten eines physischen Datenträgers

Falls der Bereich der Metadaten der Datenträgergruppe eines physischen Datenträgers aus Versehen überschrieben oder anderweitig zerstört wurde, erhalten Sie eine Fehlermeldung, die anzeigt, dass der Bereich der Metadaten inkorrekt ist, oder dass das System einen physischen Datenträger mit einer bestimmten UUID nicht finden konnte. Ggf. können Sie die Daten des physischen Datenträgers wiederherstellen, indem Sie unter Angabe derselben UUID wie bei den verlorenen Metadaten einen neuen Metadaten-Bereich auf dem physischen Datenträger erstellen.



**Warnung**

Sie sollten diese Prozedur nicht auf einem gerade aktiven, logischen LVM-Datenträger versuchen. Wenn Sie die falsche UUID angeben, werden Sie Ihre Daten verlieren.

Das folgende Beispiel zeigt in etwa die Ausgabe, die Sie im Falle eines fehlenden oder fehlerhaften Metadaten-Bereichs erhalten.

```
[root@link-07 backup]# lvs -a -o +devices
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find all physical volumes for volume group VG.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find all physical volumes for volume group VG.
  ...
```

Sie können die UUID für den physischen Datenträger, der überschrieben wurde ggf. finden, wenn Sie einen Blick in das Verzeichnis **/etc/lvm/archive** werfen. Suchen Sie in der Datei ***VolumeGroupName_xxxx*.vg** nach den letzten bekannten, archivierten LVM-Metadaten für diese Datenträgergruppe.

Alternativ können Sie die UUID des fehlenden, korrumpierten physischen Datenträgers ggf. durch das Deaktivieren des Datenträgers und Setzen des Parameters **partial** (**-P**) ermitteln.

```
[root@link-07 backup]# vgchange -an --partial
  Partial mode. Incomplete volume groups will be activated read-only.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  ...
```

Verwenden Sie die Parameter **--uuid** und **--restorefile** des Befehls **pvcreate**, um den physischen Datenträger wiederherzustellen. Das folgende Beispiel kennzeichnet das Gerät **/dev/sdh1** als einen physischen Datenträger mit der oben angezeigten UUID **FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk**. Dieser Befehl stellt das Label des physischen Datenträgers mit den in **VG_00050.vg** enthaltenen Informationen der Metadaten, also mit den aktuellsten, archivierten Metadaten für diese Datenträgergruppe, wieder her. Der Parameter **restorefile** weist den Befehl **pvcreate** an, den neuen physischen Datenträger mit dem alten in der Datenträgergruppe kompatibel zu machen, so dass sichergestellt wird, dass die neuen Metadaten nicht dort platziert werden, wo sich Daten auf dem alten physischen Datenträger befanden (was passieren kann, wenn der ursprüngliche Befehl **pvcreate** beispielsweise die Befehlszeilenparameter enthielt, die die Platzierung der Metadaten kontrollieren oder falls der physische Datenträger ursprünglich mit einer unterschiedlichen Version der Software erstellt wurde, die unterschiedliche Standardwerte verwendete). Der Befehl **pvcreate** überschreibt lediglich die LVM-Metadatenbereiche und wirkt sich nicht auf bestehende Datenbereiche aus.

```
[root@link-07 backup]# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk" --restorefile /etc/lvm/archive/VG_00050.vg /dev/sdh1
  Physical volume "/dev/sdh1" successfully created
```

Mithilfe des Befehls **vgcfgrestore** können Sie die Metadaten der Datenträgergruppe wiederherstellen.

```
[root@link-07 backup]# vgcfgrestore VG
  Restored volume group VG
```

Sie können die logischen Datenträger nun anzeigen.

```
[root@link-07 backup]# lvs -a -o +devices
  LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe VG   -wi--- 300.00G                               /dev/sdh1
(0),/dev/sda1(0)
  stripe VG   -wi--- 300.00G                               /dev/sdh1
(34728),/dev/sdb1(0)
```

Die folgenden Befehle aktivieren die Datenträger und zeigen die aktiven an.

```
[root@link-07 backup]# lvchange -ay /dev/VG/stripe
[root@link-07 backup]# lvs -a -o +devices
  LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe VG   -wi-a- 300.00G                               /dev/sdh1
(0),/dev/sda1(0)
  stripe VG   -wi-a- 300.00G                               /dev/sdh1
(34728),/dev/sdb1(0)
```

Falls die LVM-Metadaten auf der Platte mindestens genauso viel Platz einnehmen, wie das, was diese überschrieben hat, kann dieser Befehl den physischen Datenträger wiederherstellen. Falls das, was die Metadaten überschrieben hat, über den Bereich der Metadaten hinaus ging, sind davon ggf. Daten auf dem Datenträger betroffen. Mithilfe des Befehls **fsck** können Sie diese Daten ggf. wiederherstellen.

## 6.5. Ersetzen eines fehlenden physischen Datenträgers

Falls ein physischer Datenträger ausfällt oder aus anderen Gründen ersetzt werden muss, können Sie einen neuen physischen Datenträger kennzeichnen, um den ausgefallenen Datenträger in der existierenden Datenträgergruppe zu ersetzen. Hierfür können Sie die folgende Vorgehensweise befolgen, wie bei der Wiederherstellung von Metadaten eines physischen Datenträgers, wie unter Abschnitt 6.4, „Wiederherstellen von Metadaten eines physischen Datenträgers“ beschrieben. Mit den Parametern **--partial** und **--verbose** des Befehls **vgdisplay** können Sie die UUIDs und Größen von jedem beliebigen physischen Datenträger, der nicht mehr existiert, anzeigen. Falls Sie einen anderen physischen Datenträger der gleichen Größe ersetzen möchten, können Sie den Befehl **pvcreate** mit den Parametern **--restorefile** und **--uuid** verwenden, um ein neues Gerät mit derselben UUID wie die des fehlenden physischen Datenträgers zu initialisieren. Sie können dann den Befehl **vgcfgrestore** verwenden, um die Metadaten der Datenträgergruppe wiederherzustellen.

## 6.6. Entfernen von verlorenen physischen Datenträgern aus einer Datenträgergruppe

Falls Sie einen physischen Datenträger verlieren, können Sie die verbleibenden physischen Datenträger in der Datenträgergruppe mit dem Parameter **--partial** des Befehls **vgchange** aktivieren. Sie können alle logischen Datenträger, die diesen physischen Datenträger verwendet haben, mit dem Parameter **--removemissing** des Befehls **vgreduce** entfernen.

Es wird empfohlen, dass Sie den Befehl **vgreduce** mit dem Parameter **--test** ausführen, um zu verifizieren, was Sie vernichten werden.

Wie die meisten LVM-Operationen ist der Befehl **vgreduce** insofern umkehrbar, als Sie den Befehl **vgcfgrestore** umgehend verwenden können, um die Metadaten der Datenträgergruppe auf deren vorherigen Zustand zurückzusetzen. Wenn Sie beispielsweise den Parameter **--removemissing** des Befehls **vgreduce** ohne den Parameter **--test** verwendet haben und feststellen, dass Sie logische

Datenträger entfernt haben, die Sie behalten wollten, können Sie den physischen Datenträger trotzdem entfernen und ein weiteres Mal **vgcfgrestore** ausführen, um die Datenträgergruppe auf ihren vorherigen Zustand zurückzusetzen.

## 6.7. Ungenügend freie Extents für einen logischen Datenträger

Sie erhalten möglicherweise bei der Erstellung eines logischen Datenträgers die Fehlermeldung "Insufficient free extents" (nicht genügend freie Extents), auch wenn Sie basierend auf der Ausgabe der Befehle **vgdisplay** oder **vgs** der Meinung sind, über genügend Extents zu verfügen. Dies liegt daran, dass beide Befehle Zahlen auf 2 Dezimalstellen aufrunden, um eine lesbare Ausgabe zu liefern. Um die exakte Größe anzugeben, verwenden Sie die Anzahl der freien physischen Extents an Stelle von Vielfachen von Bytes, um die Größe des logischen Datenträgers zu ermitteln.

Der Befehl **vgdisplay** beinhaltet standardmäßig diese Ausgabezeile, die die freien physischen Extents anzeigt.

```
# vgdisplay
  --- Volume group ---
  ...
  Free  PE / Size       8780 / 34.30 GB
```

Alternativ können Sie die Optionen **vg_free_count** und **vg_extent_count** des Befehls **vgs** verwenden, um die freien Extents und die gesamte Summe der Extents anzuzeigen.

```
[root@tng3-1 ~]# vgs -o +vg_free_count,vg_extent_count
  VG     #PV #LV #SN Attr   VSize  VFree  Free #Ext
  testvg   2   0   0 wz--n- 34.30G 34.30G 8780 8780
```

Mit 8780 freien physischen Extents können Sie den folgenden Befehl ausführen, unter Verwendung der l-Option (kleines L), um Extents anstelle von Bytes zu verwenden:

```
# lvcreate -l8780 -n testlv testvg
```

Auf diese Weise werden alle freien Extents in der Datenträgergruppe verwendet.

```
# vgs -o +vg_free_count,vg_extent_count
  VG     #PV #LV #SN Attr   VSize  VFree Free #Ext
  testvg   2   1   0 wz--n- 34.30G    0     0 8780
```

Alternativ können Sie den logischen Datenträger auch erweitern, um einen Prozentsatz des verbleibenden freien Platzes in der Datenträgergruppe zu nutzen, indem Sie den Parameter **-l** des Befehls **lvcreate** verwendet. Werfen Sie einen Blick auf Abschnitt 4.4.1, „Lineare logische Datenträger erstellen“ für weitere Informationen.

# Kapitel 7. LVM-Administration mit dem LVM-GUI

Zusätzlich zur Befehlszeile (Command Line Interface oder kurz CLI) bietet LVM eine grafische Benutzeroberfläche (Graphical User Interface oder kurz GUI), mit der Sie logische LVM-Datenträger konfigurieren können. Sie können dieses Dienstprogramm starten, indem Sie **`system-config-lvm`** eingeben. Das LVM-Kapitel des *Handbuchs zur Speicheradministration* liefert schrittweise Anleitungen für die Konfiguration eines logischen LVM-Datenträgers mithilfe dieses Dienstprogramms.

# Der Device-Mapper

Der Device-Mapper ist ein Kernel-Treiber, der ein Framework zur Verwaltung von Datenträgern bietet. Er bietet einen generischen Weg zur Erstellung von gemappten Geräten, die ggf. als logische Datenträger verwendet werden. Er hat jedoch keine Kenntnis von Datenträgergruppen oder Metadaten-Formaten.

Der Device-Mapper liefert die Grundlage für eine Reihe von anspruchsvollen Technologien. Zusätzlich zu LVM verwenden Device-Mapper multipath und der **dmraid**-Befehl den Device-Mapper. Die Applikationsschnittstelle zum Device Mapper ist der **ioctl**-Systemaufruf. Die Benutzerschnittstelle ist der **dmsetup**-Befehl.

Logische LVM-Datenträger werden unter Verwendung des Device-Mappers aktiviert. Jeder logischer Datenträger wird in ein gemapptes Gerät übersetzt, jedem Segment wird eine Zeile in der Mapping-Tabelle zugewiesen, die das Gerät beschreibt. Der Device-Mapper unterstützt eine Vielzahl von Mapping-Zielen, unter anderem lineares Mapping, Striped Mapping und Fehler-Mapping. So können beispielsweise zwei Platten können mit einem Paar linearer Mappings pro Platte in einem logischen Datenträger zusammengefasst werden. Wenn LVM einen Datenträger erstellt, erzeugt es ein zugrunde liegendes Device-Mapper-Gerät, das mit dem **dmsetup**-Befehl abgerufen werde kann. Werfen Sie bitte einen Blick auf Abschnitt A.1, „Gerätetabelle-Mappings“ für Informationen über das Format von Geräten in einer Mapping-Tabelle. Informationen über die Verwendung des **dmsetup**-Befehls zum Abruf eines Geräts finden Sie in Abschnitt A.2, „Der dmsetup-Befehl“.

## A.1. Gerätetabelle-Mappings

Ein gemapptes Gerät ist definiert durch eine Tabelle, die spezifiziert, wie jeder Bereich logischer Sektoren auf dem Gerät mithilfe eines unterstützten Gerätetabelle-Mappings zugewiesen wird. Die Tabelle für ein gemapptes Gerät setzt sich aus einer Reihe von Zeilen im folgenden Format zusammen:

```
start length mapping [mapping_parameters...]
```

In der ersten Zeile einer Device-Mapper-Tabelle muss der ***start***-Parameter 0 entsprechen. Die ***start*** + ***length***-Parameter in einer Zeile müssen dem ***start***-Parameter in der folgenden Zeile entsprechen. Welche Mapping-Parameter in einer Zeile der Mapping-Tabelle angegeben sind, hängt davon ab, welcher ***mapping***-Typ in der Zeile spezifiziert ist.

Größen im Device-Mapper werden immer in Sektoren (512 Bytes) angegeben.

Wenn ein Gerät als ein Mapping-Parameter im Device-Mapper spezifiziert ist, kann es anhand seines Gerätenamens im Dateisystem referenziert werden (z.B. **/dev/hda**) oder anhand seiner Major- und Minor-Nummern im Format ***major***:***minor***. Das Major:Minor-Format ist dabei vorzuziehen, da auf diese Weise Pfadnamen-Aufrufe vermieden werden.

Sehen Sie nachfolgend ein Beispiel einer Mapping-Tabelle für ein Gerät. Diese Tabelle enthält vier lineare Ziele:

```
0 35258368 linear 8:48 65920
35258368 35258368 linear 8:32 65920
70516736 17694720 linear 8:16 17694976
88211456 17694720 linear 8:16 256
```

Die ersten zwei Parameter jeder Zeile stellen den Startblock und die Länge des Segments dar. Das nächste Schlüsselwort ist das Mapping-Ziel, was in allen Fällen unseren Beispiels **linear** ist. Der Rest der Zeile besteht aus den Parametern für ein **linear**-Ziel.

Die folgenden Unterabschnitte beschreiben das Format der folgenden Mappings:

- linear
- striped
- mirror
- snapshot und snapshot-origin
- error
- zero
- multipath
- crypt

## A.1.1. Das "linear" Mapping-Ziel

Ein lineares Mapping-Ziel weist einem anderen Blockgerät einen zusammenhängenden Bereich von Blöcken zu. Das Format eines linearen Ziels sieht folgendermaßen aus:

```
start length linear device offset
```

***start***

Startblock im virtuellen Gerät

***length***

Länge dieses Segments

***device***

Blockgerät, referenziert anhand des Gerätenamens im Dateisystem oder anhand der Major- und Minor-Nummern im Format ***major***:***minor***

***offset***

Start-Offset des Mappings auf dem Gerät

Das folgende Beispiel zeigt ein lineares Ziel mit dem Startblock 0 im virtuellen Gerät, einer Segmentlänge von 1638400, einem Major:Minor-Nummernpaar von 8:2, und einem Start-Offset für das Gerät von 41146992.

```
0 16384000 linear 8:2 41156992
```

Das folgende Beispiel zeigt ein lineares Ziel mit dem Geräteparameter spezifiziert als Gerät **/dev/hda**.

```
0 20971520 linear /dev/hda 384
```

## A.1.2. Das "striped" Mapping-Ziel

Das striped Mapping-Ziel unterstützt Striping (Verteilung der Daten) über physische Geräte. Es akzeptiert als Parameter die Anzahl der Stripes und die Striping-Chunk-Größe, gefolgt von einer Liste mit Gerätenamen/Sektor-Paaren. Das Format eines Striped-Ziels sieht folgendermaßen aus:

```
start length striped #stripes chunk_size device1 offset1 ... deviceN offsetN
```

Es gibt ein Set mit ***device*** und ***offset***-Parametern für jeden Stripe.

***start***
: Startblock im virtuellen Gerät

***length***
: Länge dieses Segments

***#stripes***
: Anzahl der Stripes für das virtuelle Gerät

***chunk_size***
: Anzahl der Sektoren, die auf jeden Stripe geschrieben werden, bevor zum nächsten gewechselt wird; muss eine Zweierpotenz sein, die mindestens so groß wie die Kernel-Seitengröße ist

***device***
: Blockgerät, referenziert anhand des Gerätenamens im Dateisystem oder anhand der Major- und Minor-Nummern im Format ***major***:***minor***

***offset***
: Start-Offset des Mappings auf dem Gerät

Das folgende Beispiel zeigt ein Striped-Ziel mit drei Stripes und einer Chunk-Größe von 128:

```
0 73728 striped 3 128 8:9 384 8:8 384 8:7 9789824
```

**0**
: Startblock im virtuellen Gerät

**73728**
: Länge dieses Segments

**striped 3 128**
: Striping über drei Geräte mit einer Chunk-Größe von 128 Blöcken

**8:9**
: Major:Minor-Nummern des ersten Geräts

**384**
: Start-Offset des Mappings auf dem ersten Gerät

**8:8**
: Major:Minor-Nummern des zweiten Geräts

**384**
Start-Offset des Mappings auf dem zweiten Gerät

**8:7**
Major:Minor-Nummern des dritten Geräts

**9789824**
Start-Offset des Mappings auf dem dritten Gerät

Das folgende Beispiel zeigt ein Striped-Ziel mit zwei Stripes und einer Chunk-Größe von 256 KiB, mit den Geräteparametern spezifiziert durch die Gerätenamen im Dateisystemen anstelle der Major- und Minor-Nummern:

```
0 65536 striped 2 512 /dev/hda 0 /dev/hdb 0
```

### A.1.3. Das "mirror" Mapping-Ziel

Das "mirror" Mapping-Ziel unterstützt das Mapping eines gespiegelten logischen Geräts. Das Format eines Mirrored-Ziels sieht folgendermaßen aus:

```
start length mirror log_type #logargs logarg1 ... logargN #devs device1 offset1
... deviceN offsetN
```

***start***
Startblock im virtuellen Gerät

***length***
Länge dieses Segments

***log_type***
Die möglichen Protokolltypen und deren Parameter lauten wie folgt:

**core**
Der Mirror ist lokal und das Mirror-Protokoll wird im Kernspeicher bewahrt. Dieser Protokolltyp akzeptiert 1 - 3 Parameter:

***regionsize*** [[**no**]**sync**] [**block_on_error**]

**disk**
Der Mirror ist lokal und das Mirror-Protokoll wird auf der Festplatte bewahrt. Dieser Protokolltyp akzeptiert 2 - 4 Parameter:

***logdevice regionsize*** [[**no**]**sync**] [**block_on_error**]

**clustered_core**
Der Mirror ist geclustert und das Mirror-Protokoll wird im Kernspeicher bewahrt. Dieser

Protokolltyp akzeptiert 2 - 4 Parameter:

***regionsize UUID*** [[**no**]**sync**] [**block_on_error**]

**clustered_disk**

Der Mirror ist geclustert und das Mirror-Protokoll wird auf der Festplatte bewahrt. Dieser Protokolltyp akzeptiert 3 - 5 Parameter:

***logdevice regionsize UUID*** [[**no**]**sync**] [**block_on_error**]

LVM pflegt eine kleine Protokolldatei, in der festgehalten wird, welche Bereiche mit dem (den) Mirror(s) synchron sind. Der ***regionsize***-Parameter bestimmt die Größe dieser Bereiche.

In einer geclusterten Umgebung ist der ***UUID***-Parameter ein eindeutiger Bezeichner, der dem Mirror-Protokollgerät zugewiesen ist, so dass der Protokollzustand über den Cluster hinweg bewahrt werden kann.

Der optionale **[no]sync**-Parameter kann dazu verwendet werden, den Mirror als "in-sync" oder "out-of-sync" zu spezifizieren. Mithilfe des **block_on_error**-Parameters wird der Mirror angewiesen, auf Fehler zu antworten statt diese zu ignorieren.

***#log_args***

Anzahl der Protokollparameter, die im Mapping spezifiziert werden.

***logargs***

Die Protokollparameter für den Mirror; die Anzahl der angegebenen Protokollparameter wird durch den ***#log-args***-Parameter spezifiziert und die gültigen Protokollparameter werden durch den ***log_type***-Parameter bestimmt.

***#devs***

Die Anzahl der Standbeine des Mirrors; ein Gerät und ein Offset wird für jedes Standbein spezifiziert.

***device***

Blockgerät für jedes Mirror-Standbein, referenziert anhand des Gerätenamens im Dateisystem oder anhand der Major- und Minor-Nummern im Format ***major***:***minor***. Ein Blockgerät und ein Offset wird für jedes Standbein spezifiziert, das vom ***#devs***-Parameter angegeben ist.

***offset***

Start-Offset des Mappings auf dem Gerät. Ein Blockgerät und ein Offset wird für jedes Standbein spezifiziert, das vom ***#devs***-Parameter angegeben ist.

Das folgende Beispiel zeigt ein Mirror-Mapping-Ziel für einen geclusterten Mirror mit einem auf der Festplatte bewahrten Mirror-Protokoll.

```
0 52428800 mirror clustered_disk 4 253:2 1024 UUID block_on_error 3 253:3 0 253:4
0 253:5 0
```

**0**
: Startblock im virtuellen Gerät

**52428800**
: Länge dieses Segments

**mirror clustered_disk**
: Mirror-Ziel mit einem Protokolltyp, der spezifiziert, dass der Mirror gespiegelt wird und das Mirror-Protokoll auf Festplatte bewahrt wird

**4**
: 4 Mirror-Protokollparameter folgen

**253:2**
: Major:Minor-Nummern des Protokollgeräts

**1024**
: Bereichsgröße, die das Mirror-Protokoll verwendet um nachzuverfolgen, was synchron ist

***UUID***
: UUID des Mirror-Protokollgeräts, um Protokolldaten für den gesamten Cluster zu bewahren

**block_on_error**
: Mirror sollte auf Fehler antworten

**3**
: Anzahl der Standbeine im Mirror

**253:3 0 253:4 0 253:5 0**
: Major:Minor-Nummern und Offset für die Geräte, aus denen jedes Standbein des Mirrors besteht

### A.1.4. Snapshot und Snapshot-Quelle der Mapping-Ziele

Wenn Sie den ersten LVM-Snapshot eines Datenträgers erstellen, werden vier Device-Mapper-Geräte verwendet:

1. Ein Gerät mit einem **linear** Mapping, das die ursprüngliche Mapping-Tabelle des Quelldatenträgers enthält.
2. Ein Gerät mit einem **linear** Mapping, das als "copy-on-write" (kurz COW) Gerät für den Quelldatenträger dient; bei jedem Schreibvorgang werden die originalen Daten im COW-Gerät eines jeden Snapshots gespeichert, um dessen sichtbaren Inhalt unverändert zu lassen (bis das

COW-Gerät voll ist).

3. Ein Gerät mit einem **snapshot**-Mapping, das #1 und #2 kombiniert, was den sichtbaren Snapshot-Datenträger bildet.
4. Der "Original"-Datenträger (der die Gerätenummer des originalen Quelldatenträgers verwendet), dessen Tabelle durch ein "Snapshot-Quell"-Mapping von Gerät #1 ersetzt wird.

Bei der Erstellung dieser Geräte wird ein festes Namensschema verwendet, z.B. können Sie die folgenden Befehle verwenden, um einen LVM-Datenträger namens **base** und einen Snapshot-Datenträger namens **snap** basierend auf diesem Datenträger zu erstellen.

```
# lvcreate -L 1G -n base volumeGroup
# lvcreate -L 100M --snapshot -n snap volumeGroup/base
```

Daraus entstehen vier Geräte, die Sie sich mit dem folgenden Befehl anzeigen lassen können:

```
# dmsetup table|grep volumeGroup
volumeGroup-base-real: 0 2097152 linear 8:19 384
volumeGroup-snap-cow: 0 204800 linear 8:19 2097536
volumeGroup-snap: 0 2097152 snapshot 254:11 254:12 P 16
volumeGroup-base: 0 2097152 snapshot-origin 254:11

# ls -lL /dev/mapper/volumeGroup-*
brw-------  1 root root 254, 11 29 ago 18:15 /dev/mapper/volumeGroup-base-real
brw-------  1 root root 254, 12 29 ago 18:15 /dev/mapper/volumeGroup-snap-cow
brw-------  1 root root 254, 13 29 ago 18:15 /dev/mapper/volumeGroup-snap
brw-------  1 root root 254, 10 29 ago 18:14 /dev/mapper/volumeGroup-base
```

Das Format des **snapshot-origin**-Ziels lautet folgendermaßen:

```
start length snapshot-origin origin
```

***start***
: Startblock im virtuellen Gerät

***length***
: Länge dieses Segments

***origin***
: Basisdatenträger des Snapshots

Auf dem **snapshot-origin** basieren normalerweise ein oder mehrere weitere Snapshots. Lesevorgänge werden direkt auf das dahinter liegende Gerät gemappt. Bei jedem Schreibvorgang werden die originalen Daten im COW-Gerät eines jeden Snapshots gespeichert, um dessen sichtbaren Inhalt unverändert zu lassen, bis das COW-Gerät voll ist.

Das Format des **snapshot**-Ziels sieht folgendermaßen aus:

```
start length snapshot origin COW-device P|N chunksize
```

***start***

Startblock im virtuellen Gerät

***length***
Länge dieses Segments

***origin***
Basisdatenträger des Snapshots

***COW-device***
Gerät, auf dem veränderte Datenblöcke gespeichert werden

**P|N**
P (Persistent) oder N (Nicht persistent); gibt an, ob ein Snapshot über einen Neustart hinweg bestehen bleibt. Für transiente Snapshots (N) müssen weniger Metadaten auf der Festplatte gespeichert werden; sie können vom Kernel im Hauptspeicher bewahrt werden.

***chunksize***
Größe in Sektoren der veränderten Datenblöcke ("Chunks"), die auf dem COW-Gerät gespeichert werden

Das folgende Beispiel zeigt ein **snapshot-origin**-Ziel mit dem originalen Gerät 254:11.

```
0 2097152 snapshot-origin 254:11
```

Das folgende Beispiel zeigt ein **snapshot**-Ziel mit dem originalen Gerät 254:11 und dem COW-Gerät 254:12. Dieses Snapshot-Gerät ist über Neustarts hinweg persistent und die Chunk-Größe für die auf dem COW-Gerät gespeicherten Daten beträgt 16 Sektoren.

```
0 2097152 snapshot 254:11 254:12 P 16
```

### A.1.5. Das "error" Mapping-Ziel

Mit einem "error" Mapping-Ziel schlägt jede I/O-Operation auf dem gemappten Sektor fehl.

Ein Error-Mapping-Ziel kann zu Testzwecken eingesetzt werden. Um zu testen, wie sich ein Gerät im Fehlerfall verhält, können Sie ein Geräte-Mapping mit einem fehlerhaften Sektor in der Mitte des Geräts erstellen, oder Sie können ein Standbein eines Mirrors auslagern und dieses Standbein durch ein Error-Ziel ersetzen.

Ein Error-Ziel kann anstelle eines fehlerhaften Geräts verwendet werden, um Zeitüberschreitungen und Neuversuche auf dem eigentlichen Gerät zu vermeiden. Es kann als Zwischenziel dienen, während Sie die LVM-Metadaten bei Ausfällen neu anordnen.

Das **error** Mapping-Ziel akzeptiert außer den ***start*** und ***length***-Parametern keine weiteren Parameter.

Das folgende Beispiel zeigt ein **error**-Ziel.

```
0 65536 error
```

### A.1.6. Das "zero" Mapping-Ziel

Das **zero** Mapping-Ziel ist ein Blockgerät-Äquivalent zu **/dev/zero**. Eine Leseoperation auf diesem Mapping gibt Blöcke von Nullen zurück. Auf dieses Mapping geschriebene Daten werden verworfen, der Schreibvorgang ist jedoch erfolgreich. Das **zero** Mapping-Ziel akzeptiert außer den ***start*** und ***length***-Parametern keine weiteren Parameter.

Das folgende Beispiel zeigt ein **zero**-Ziel für ein 16 TB Gerät.

```
0 65536 zero
```

### A.1.7. Das "multipath" Mapping-Ziel

Das "multipath" Mapping-Ziel unterstützt das Mapping eines Multipath-Geräts. Das Format eines **multipath**-Ziels sieht folgendermaßen aus:

```
start length  multipath  #features [feature1 ... featureN] #handlerargs
[handlerarg1 ... handlerargN] #pathgroups pathgroup pathgroupargs1 ...
pathgroupargsN
```

Es gibt ein Set mit ***pathgroupargs***-Parametern für jede Pfadgruppe.

***start***
: Startblock im virtuellen Gerät

***length***
: Länge dieses Segments

***#features***
: Die Anzahl von Multipath-Features, gefolgt von diesen Features. Ist dieser Parameter Null, gibt es keinen ***feature***-Parameter und der nächste Geräte-Mapping-Parameter ist ***#handlerargs***. Derzeit gibt es ein unterstütztes Multipath-Feature, **queue_if_no_path**. Dies gibt an, dass dieses Multipath-Gerät derzeit darauf eingestellt ist, I/O-Operationen in eine Warteschlange zu stellen, wenn kein Pfad verfügbar ist.

  Ist beispielsweise die **no_path_retry**-Option in der **multipath.conf**-Datei darauf eingestellt, I/O-Operationen nur solange in eine Warteschlange zu stellen, bis alle Pfade als fehlerhaft markiert wurden, nachdem eine festgelegte Anzahl an Versuchen unternommen wurde, diese Pfade zu verwenden, würde das Mapping folgendermaßen erscheinen, bis für alle Pfadüberprüfungen die festgelegte Anzahl von Prüfungen fehlschlugen.

  ```
  0 71014400 multipath 1 queue_if_no_path 0 2 1 round-robin 0 2 1 66:128 \
  1000 65:64 1000 round-robin 0 2 1 8:0 1000 67:192 1000
  ```

  Nachdem für alle Pfadüberprüfungen die festgelegte Anzahl von Prüfungen fehlschlugen, würde das Mapping folgendermaßen erscheinen.

  ```
  0 71014400 multipath 0 0 2 1 round-robin 0 2 1 66:128 1000 65:64 1000 \
  round-robin 0 2 1 8:0 1000 67:192 1000
  ```

***#handlerargs***

Die Anzahl von Hardware-Handler-Parametern, gefolgt von diesen Parametern. Ein Hardware-Handler spezifiziert ein Modul, das zur Durchführung von hardwarespezifischen Aktionen verwendet wird beim Wechseln von Pfadgruppen oder bei der Handhabung von I/O-Fehlern. Ist dies auf 0 gesetzt, ist der nächste Parameter ***#pathgroups***.

***#pathgroups***

Die Anzahl von Pfadgruppen. Eine Pfadgruppe ist die Gruppe von Pfaden, über die ein Multipath-Gerät lastverteilt. Es gibt eine Gruppe von ***pathgroupargs***-Parametern für jede Pfadgruppe.

***pathgroup***

Die nächste zu versuchende Pfadgruppe.

***pathgroupsargs***

Jede Pfadgruppe umfasst die folgenden Parameter:

```
pathselector #selectorargs #paths #pathargs device1 ioreqs1 ... deviceN
ioreqsN
```

Es gibt eine Gruppe von Pfadparametern für jeden Pfad in der Pfadgruppe.

***pathselector***

Spezifiziert den verwendeten Algorithmus um zu bestimmen, welcher Pfad in dieser Pfadgruppe für die nächste I/O-Operation zu verwenden ist.

***#selectorargs***

Die Anzahl der Pfadauswahl-Parameter, die diesem Parameter im Multipath-Mapping folgen. Derzeit ist der Wert dieses Parameters immer 0.

***#paths***

Die Anzahl von Pfaden in dieser Pfadgruppe.

***#pathargs***

Die Anzahl der Pfadparameter, die für jeden Pfad in dieser Gruppe spezifiziert sind. Derzeit ist diese Anzahl immer 1, der ***ioreqs***-Parameter.

***device***

Die Blockgerätenummer des Pfads, referenziert durch die Major- und Minor-Nummern im Format ***major***:***minor***

***ioreqs***

Die Anzahl von I/O-Anfragen, die auf diesen Pfad geleitet werden, bevor zum nächsten Pfad in der aktuellen Gruppe gewechselt wird.

Abbildung A.1, „"Multipath" Mapping-Ziel“ veranschaulicht das Format eines Multipath-Ziels mit zwei Pfadgruppen.

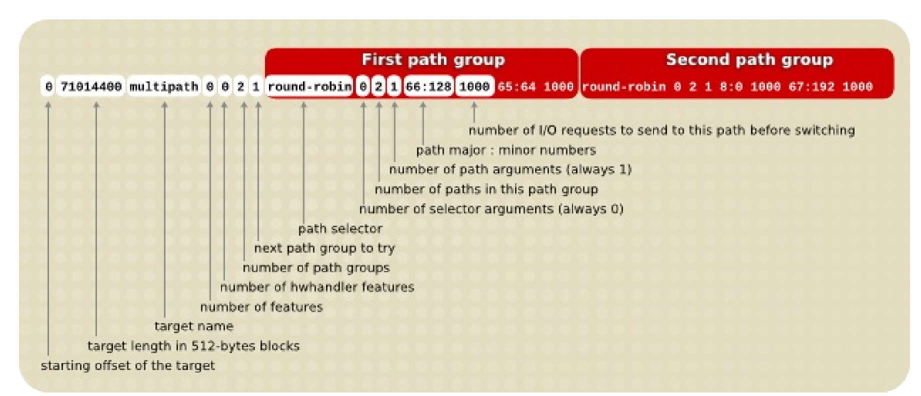



**Abbildung A.1. "Multipath" Mapping-Ziel**

Das folgende Beispiel zeigt eine reine Failover-Zieldefinition für dasselbe Multipath-Gerät. In diesem Ziel gibt es vier Pfadgruppen mit je einem offenen Pfad pro Pfadgruppe, so dass das Multipath-Gerät zu jeder Zeit nur einen Pfad verwendet.

```
0 71014400 multipath 0 0 4 1 round-robin 0 1 1 66:112 1000 \
round-robin 0 1 1 67:176 1000 round-robin 0 1 1 68:240 1000 \
round-robin 0 1 1 65:48 1000
```

Das folgende Beispiel zeigt eine vollständig verteilte (multibus) Zieldefinition für dasselbe Multipath-Gerät. In diesem Ziel gibt es nur eine Pfadgruppe, die alle Pfade umfasst. Bei diesem Aufbau verteilt Multipath die Last gleichmäßig über alle Pfade.

```
0 71014400 multipath 0 0 1 1 round-robin 0 4 1 66:112 1000 \
 67:176 1000 68:240 1000 65:48 1000
```

Weitere Informationen über den Einsatz von Multipath finden Sie im Dokument *Verwendung von Device Mapper Multipath*.

### A.1.8. Das "crypt" Mapping-Ziel

Das **crypt**-Ziel verschlüsselt die Daten, die das angegebene Gerät durchlaufen. Es verwendet die Kernel-Crypto-API.

Das Format für das **crypt**-Ziel sieht folgendermaßen aus:

```
start length crypt cipher key IV-offset device offset
```

***start***
: Startblock im virtuellen Gerät

***length***
: Länge dieses Segments

***cipher***

Cipher (Schlüssel) besteht aus ***cipher[-chainmode]-ivmode[:iv options]***.

***cipher***

Die verfügbaren Cipher (Schlüssel) sind in **/proc/crypto** aufgelistet (z.B. **aes**).

***chainmode***

Verwenden Sie stets **cbc**. Verwenden Sie nicht **ebc**, da es keinen initialen Vektor (IV) einsetzt.

***ivmode[:iv options]***

IV ist ein initialer Vektor, der zum Variieren der Verschlüsselung eingesetzt wird. Der IV-Modus ist **plain** oder **essiv:hash**. Der ***ivmode*** **-plain** verwendet die Sektorennummer (zuzüglich IV-Offset) als den IV. Der ***ivmode*** **-essiv** ist eine Verbesserung, um mögliche Wasserzeichenangriffe zu vermeiden.

***key***

Verschlüsselungs-Code, angegeben in hex

***IV-offset***

Offset für den initialen Vektor (IV)

***device***

Blockgerät, referenziert anhand des Gerätenamens im Dateisystem oder anhand der Major- und Minor-Nummern im Format ***major***:***minor***

***offset***

Start-Offset des Mappings auf dem Gerät

Im Folgenden sehen Sie ein Beispiel für ein **crypt**-Ziel.

```
0 2097152 crypt aes-plain 0123456789abcdef0123456789abcdef 0 /dev/hda 0
```

## A.2. Der dmsetup-Befehl

Der Befehl **dmsetup** ist ein Kommandozeilen-Wrapper für die Kommunikation mit dem Device-Mapper. Für allgemeine Systeminformationen zu LVM-Geräten sind die **info**, **ls**, **status** und **deps** Optionen des **dmsetup**-Befehls ggf. für Sie nützlich, wie in den nachfolgenden Abschnitten erläutert.

Werfen Sie einen Blick auf die Handbuchseite (8) von **dmsetup** für weitere Informationen zu den Optionen und Fähigkeiten des Befehls **dmsetup**.

### A.2.1. Der dmsetup info Befehl

Der **dmsetup info *device***-Befehl liefert eine Übersicht über Device-Mapper-Geräte. Wenn Sie keinen Gerätenamen angeben, werden Informationen für alle derzeit konfigurierten Device-Mapper-Geräte ausgegeben. Wenn Sie ein bestimmtes Gerät angeben, gibt dieser Befehl nur Informationen für dieses Gerät aus.

Der **dmsetup info**-Befehl liefert Informationen in den folgenden Kategorien:

**Name**

Der Name des Geräts. Ein LVM-Gerät wird durch den Datengruppennamen und den Namen des logischen Datenträgers, mit einem Bindestrich voneinander getrennt, dargestellt. Ein Bindestrich im ursprünglichen Namen wird in zwei Bindestriche übersetzt.

**State**

Die möglichen Gerätezustände sind **SUSPENDED**, **ACTIVE** und **READ-ONLY**. Der **dmsetup suspend**-Befehl setzt den Gerätezustand auf **SUSPENDED**. Ist ein Gerät ausgesetzt ("suspended"), werden sämtliche I/O-Operationen auf diesem Gerät ausgesetzt. Der **dmsetup resume**-Befehl setzt den Gerätezustand zurück auf **ACTIVE**.

**Read Ahead**

Die Anzahl von Datenblöcken, die das System im Voraus liest für jede offene Datei, auf der Leseoperationen stattfinden. Standardmäßig wählt der Kernel automatisch einen angemessenen Wert. Sie können diesen Wert mithilfe der **--readahead**-Option des **dmsetup**-Befehls ändern.

**Tables present**

Mögliche Zustände für diese Kategorie sind **LIVE** und **INACTIVE**. Der Zustand **INACTIVE** zeigt an, dass eine Tabelle geladen wurde, die vom Swap-Speicher in den Arbeitsspeicher geladen wird, sobald ein **dmsetup resume**-Befehl einen Gerätezustand wieder auf **ACTIVE** setzt, woraufhin sich der Tabellenzustand auf **LIVE** ändert. Weitere Informationen diesbezüglich finden Sie auf der **dmsetup**-Handbuchseite.

**Open count**

Der Zähler für offene Referenzen gibt an, wie oft das Gerät geöffnet ist. Ein **mount**-Befehl öffnet ein Gerät.

**Event number**

Die derzeitige Anzahl empfangener Ereignisse. Durch Ausführen des Befehls **dmsetup wait *n*** kann ein Benutzer auf das n-te Ereignis warten, und den Aufruf solange blockieren, bis dies empfangen wurde.

**Major, minor**

Major- und Minor-Gerätenummer

**Number of targets**

Die Anzahl der Fragmente, aus denen sich ein Gerät zusammensetzt. So hat zum Beispiel ein lineares Gerät, das 3 Festplatten umfasst, 3 Ziele. Ein lineares Gerät, das aus dem Anfang und dem Ende einer Platte besteht, jedoch nicht der Mitte, hätte 2 Ziele.

**UUID**
: UUID des Geräts.

Das folgende Beispiel zeigt einen Ausschnitt der Ausgabe des **dmsetup info**-Befehls.

```
[root@ask-07 ~]# dmsetup info
Name:              testgfsvg-testgfslv1
State:             ACTIVE
Read Ahead:        256
Tables present:    LIVE
Open count:        0
Event number:      0
Major, minor:      253, 2
Number of targets: 2
UUID: LVM-K528WUGQgPadNXYcFrrf9LnPlUMswgkCkpgPIgYzSvigM7SfeWCypddNSWtNzc2N
...
Name:              VolGroup00-LogVol00
State:             ACTIVE
Read Ahead:        256
Tables present:    LIVE
Open count:        1
Event number:      0
Major, minor:      253, 0
Number of targets: 1
UUID: LVM-tOcS1kqFV9drb0X1Vr8sxeYP0tqcrpdegyqj5lZxe45JMGlmvtqLmbLpBcenh2L3
```

### A.2.2. Der dmsetup ls Befehl

Mithilfe des **dmsetup ls**-Befehls können Sie die Gerätenamen gemappter Geräte anzeigen. Sie können speziell Geräte anzeigen lassen, die mindestens ein Ziel vom angegebenen Typ haben, indem Sie den Befehl **dmsetup ls --target *target_type*** ausführen. Werfen Sie für weitere Optionen des **dmsetup ls**-Befehls einen Blick auf die **dmsetup**-Handbuchseite.

Das folgende Beispiel zeigt den Befehl, um die Gerätenamen der derzeit konfigurierten gemappten Geräte anzuzeigen.

```
[root@ask-07 ~]# dmsetup ls
testgfsvg-testgfslv3    (253, 4)
testgfsvg-testgfslv2    (253, 3)
testgfsvg-testgfslv1    (253, 2)
VolGroup00-LogVol01     (253, 1)
VolGroup00-LogVol00     (253, 0)
```

Das folgende Beispiel zeigt den Befehl, um die Gerätenamen der derzeit konfigurierten Mirror-Mappings anzuzeigen.

```
[root@grant-01 ~]# dmsetup ls --target mirror
lock_stress-grant--02.1722      (253, 34)
lock_stress-grant--01.1720      (253, 18)
lock_stress-grant--03.1718      (253, 52)
lock_stress-grant--02.1716      (253, 40)
lock_stress-grant--03.1713      (253, 47)
lock_stress-grant--02.1709      (253, 23)
lock_stress-grant--01.1707      (253, 8)
lock_stress-grant--01.1724      (253, 14)
lock_stress-grant--03.1711      (253, 27)
```

LVM-Konfigurationen, die auf Multipath oder anderen Device-Mapper-Geräten aufbauen, können unübersichtlich sein. Der Befehl **`dmsetup ls`** bietet eine **`--tree`**-Option, die Abhängigkeiten zwischen Geräten in Baumstruktur anzeigt, wie im folgenden Beispiel veranschaulicht.

```
# dmsetup ls --tree
vgtest-lvmir (253:13)
 ├─vgtest-lvmir_mimage_1 (253:12)
 │  └─mpathep1 (253:8)
 │     └─mpathe (253:5)
 │        ├─ (8:112)
 │        └─ (8:64)
 ├─vgtest-lvmir_mimage_0 (253:11)
 │  └─mpathcp1 (253:3)
 │     └─mpathc (253:2)
 │        ├─ (8:32)
 │        └─ (8:16)
 └─vgtest-lvmir_mlog (253:4)
    └─mpathfp1 (253:10)
       └─mpathf (253:6)
          ├─ (8:128)
          └─ (8:80)
```

## A.2.3. Der dmsetup status Befehl

Der **`dmsetup status`** ***`device`*** Befehl liefert Zustandsinformationen für jedes Ziel im angegebenen Gerät. Wenn Sie keinen Gerätenamen angeben, werden Informationen für alle derzeit konfigurierten Device-Mapper-Geräte ausgegeben. Sie können den Status von ausschließlich jenen Geräten anzeigen lassen, die mindestens ein Ziel vom angegebenen Typ haben, indem Sie den Befehl **`dmsetup status --target`** ***`target_type`*** ausführen.

Das folgende Beispiel zeigt den Befehl, um die Zustände von Zielen in allen derzeit konfigurierten gemappten Geräten anzuzeigen.

```
[root@ask-07 ~]# dmsetup status
testgfsvg-testgfslv3: 0 312352768 linear
testgfsvg-testgfslv2: 0 312352768 linear
testgfsvg-testgfslv1: 0 312352768 linear
testgfsvg-testgfslv1: 312352768 50331648 linear
VolGroup00-LogVol01: 0 4063232 linear
VolGroup00-LogVol00: 0 151912448 linear
```

## A.2.4. Der dmsetup deps Befehl

Der **`dmsetup deps`** ***`device`*** Befehl liefert eine Liste mit (Major,Minor)-Paaren für Geräte, die von der Mapping-Tabelle des spezifizierten Geräts referenziert werden. Wenn Sie keinen Gerätenamen angeben, werden Informationen für alle derzeit konfigurierten Device-Mapper-Geräte ausgegeben.

Das folgende Beispiel zeigt den Befehl, um die Abhängigkeiten für alle derzeit konfigurierten Device-Mapper-Geräte anzuzeigen.

```
[root@ask-07 ~]# dmsetup deps
testgfsvg-testgfslv3: 1 dependencies    : (8, 16)
testgfsvg-testgfslv2: 1 dependencies    : (8, 16)
testgfsvg-testgfslv1: 1 dependencies    : (8, 16)
VolGroup00-LogVol01: 1 dependencies     : (8, 2)
VolGroup00-LogVol00: 1 dependencies     : (8, 2)
```

Das folgende Beispiel zeigt den Befehl, um die Abhängigkeiten nur für das Gerät **lock_stress-grant--02.1722** anzuzeigen.

```
[root@grant-01 ~]# dmsetup deps lock_stress-grant--02.1722
3 dependencies  : (253, 33) (253, 32) (253, 31)
```

## A.3. Device-Mapper-Unterstützung für den udev-Gerätemanager

Die hauptsächliche Aufgabe des **udev**-Gerätemanagers besteht darin, einen dynamischen Weg zum Erzeugen von Knoten im **/dev**-Verzeichnis bereitzustellen. Das Erzeugen dieser Knoten wird durch die Anwendung von **udev**-Regeln im Userspace geleitet. Diese Regeln werden auf **udev**-Ereignisse angewendet, die direkt vom Kernel gesendet werden als Reaktion auf das Hinzufügen, Entfernen oder Ändern bestimmter Geräte. Somit können auf bequeme und zentral Weise Geräte im laufenden Betrieb verändert werden.

Neben der Erstellung der Knoten selbst kann der **udev**-Gerätemanager auch symbolische Links mit jeweils eigenen Namen erzeugen, so dass es Benutzern freisteht, bei Bedarf ihre eigene Namens- und Verzeichnisstruktur im **/dev**-Verzeichnis zu wählen.

Jedes **udev**-Ereignis enthält wesentliche Informationen über das bearbeitete Gerät, u.a. dessen Name, das Subsystem zu dem es gehört, sein Gerätetyp, seine verwendete Major- und Minor-Nummer und die Art des Ereignisses. Dank dieser Informationen sowie dank der Möglichkeit, auf alle Informationen innerhalb des **/sys**-Verzeichnisses zuzugreifen, auf das auch innerhalb der **udev**-Regeln zugegriffen werden kann, können Benutzer einfache Filter basierend auf diesen Informationen einsetzen und die Regeln abhängig von diesen Informationen bedingt ausführen.

Der **udev**-Gerätemanager bietet zudem einen zentralen Weg zum Einrichten der Knotenberechtigungen. Ein Benutzer kann einfach eine angepasste Reihe von Regeln hinzufügen, um die Berechtigungen für jedes Gerät zu definieren, das durch eine Information, die während der Verarbeitung des Ereignisses zur Verfügung steht, spezifiziert wird.

Es ist außerdem möglich, Programm-Hooks direkt in **udev**-Regeln hinzuzufügen. Der **udev**-Gerätemanager kann diese Programme aufrufen, um eine weitergehende Verarbeitung zu veranlassen, die zur Handhabung dieses Ereignisses notwendig ist. Zudem kann das Programm als Ergebnis dieser Verarbeitung Umgebungsvariablen exportieren. Jegliche gelieferte Ergebnisse können im weiteren Verlauf als weitere Informationsquelle in den Regeln genutzt werden.

Jede Software, die die **udev**-Bibliothek verwendet, kann **udev**-Ereignisse mit allen verfügbaren Informationen empfangen und verarbeiten, die Verarbeitung ist also nicht allein an den **udev**-Daemon gebunden.

### A.3.1. udev-Integration mit dem Device Mapper

In RHEL 6 bietet der Device Mapper direkte Unterstützung für die **udev**-Integration. Dies synchronisiert den Device Mapper mit sämtlicher **udev**-Verarbeitung in Zusammenhang mit Device-Mapper-Geräten, einschließlich LVM-Geräten. Diese Synchronisation ist notwendig, da die Regelanwendung im **udev**-Daemon eine Art paralleler Verarbeitung ist zu dem Programm, das die Quelle für die Änderungen an diesem Gerät darstellt (z.B. **dmsetup** und LVM). Ohne diese Unterstützung gab es häufig das Problem, dass ein Benutzer versuchte, ein Gerät zu entfernen, das noch geöffnet war und aufgrund eines vorangegangenen Änderungsereignisses immer noch von **udev**-Regeln bearbeitet wurde; dies trat insbesondere dann auf, wenn die Zeitabstände zwischen Änderungen auf diesem Gerät sehr kurz waren.

Die RHEL 6 Release bietet offiziell unterstützte **udev**-Regeln für Device-Mapper-Geräte im Allgemeinen und für LVM im Besonderen. Tabelle A.1, „udev-Regeln für Device-Mapper-Geräte“ fasst diese Regeln zusammen, die alle in **/lib/udev/rules.d** installiert sind.

**Tabelle A.1. udev-Regeln für Device-Mapper-Geräte**

| Dateiname | Beschreibung |
|---|---|
| **10-dm.rules** | Enthält grundlegende/allgemeine Device-Mapper-Regeln und erstellt die symbolischen Links in **/dev/mapper** mit einem **/dev/dm-*N***-Ziel, wobei N eine dem Gerät durch den Kernel dynamisch zugewiesene Nummer ist (**/dev/dm-*N*** ist ein Knoten).<br><br>ANMERKUNG: **/dev/dm-*N***-Knoten sollten *niemals* in Skripten zum Zugriff auf das Gerät verwendet werden, da die Nummer N dynamisch zugewiesen wird und sich abhängig von der Reihenfolge, in der die Geräte aktiviert werden, ändern kann. Daher sollten echte Namen im **/dev/mapper**-Verzeichnis verwendet werden. Dieser Aufbau unterstützt **udev**-Anforderungen hinsichtlich der Erstellung von Knoten/Symlinks. |
| **11-dm-lvm.rules** | Enthält auf LVM-Geräte angewendete Regeln und erstellt die symbolischen Links für die logischen Datenträger der Datenträgergruppe. Die symbolischen Links werden im **/dev/*vgname***-Verzeichnis mit einem **/dev/dm-*N***-Ziel erstellt.<br><br>ANMERKUNG: Um bei der Benennung aller zukünftigen Regeln für Device-Mapper-Untersysteme konsistent mit dem Standard zu bleiben, sollten udev-Regeln dem Format **11-dm-*subsystem_name*.rules** entsprechen. Auch alle **libdevmapper**-Benutzer, die ebenfalls **udev**-Regeln erstellen, sollten diesem Standard folgen. |
| **13-dm-disk.rules** | Enthält auf alle Device-Mapper-Geräte allgemein angewendete Regeln und erstellt die symbolischen Links in den **/dev/disk/by-id**, **/dev/disk/by-uuid** und **/dev/disk/by-uuid**-Verzeichnissen. |
| **95-dm-notify.rules** | Enthält die Regel zur Benachrichtigung des wartenden Prozesses mittels **libdevmapper** (ganz wie LVM und **dmsetup**). Die Benachrichtigung erfolgt, nachdem alle vorhergehenden Regeln angewendet wurden, um sicherzustellen, dass jegliche **udev**-Verarbeitung abgeschlossen ist. Der benachrichtigte Prozess wird anschließend fortgesetzt. |

Sie können mithilfe der **12-dm-permissions.rules**-Datei zusätzliche, angepasste Berechtigungsregeln hinzufügen. Diese Datei ist *nicht* im **/lib/udev/rules**-Verzeichnis installiert; sie befindet sich stattdessen im **/usr/share/doc/device-mapper-*version***-Verzeichnis. Die **12-dm-permissions.rules**-Datei ist eine Vorlage, die Hinweise zum Einstellen der Berechtigungen enthält, basierend auf einigen passenden Regeln als Beispiel; die Datei enthält Beispiele für einige häufige Situationen. Sie können diese Datei bearbeiten und sie manuell in das **/etc/udev/rules.d**-Verzeichnis ablegen, wo sie Aktualisierungen überdauert, die Einstellungen bleiben also erhalten.

Diese Regeln setzen alle grundlegenden Variablen, die von allen anderen Regeln bei der Verarbeitung von Ereignissen verwendet werden können.

Die folgenden Variablen werden in 10-dm.rules gesetzt:

- **DM_NAME**: Name des Device-Mapper-Geräts
- **DM_UUID**: UUID des Device-Mapper-Geräts
- **DM_SUSPENDED**: der ausgesetzte Zustand des Device-Mapper-Geräts
- **DM_UDEV_RULES_VSN**: **udev**-Regelversion (primär für alle anderen Regeln zur Überprüfung, ob zuvor erwähnte Variablen direkt von offiziellen Device-Mapper-Regeln gesetzt wurden)

Die folgenden Variablen werden in **11-dm-lvm.rules** gesetzt:

- **DM_LV_NAME**: Name des logischen Datenträgers
- **DM_VG_NAME**: Name der Datenträgergruppe
- **DM_LV_LAYER**: Name der LVM-Schicht

All diese Variablen können in der **12-dm-permissions.rules**-Datei genutzt werden, um Berechtigungen für bestimmte Device-Mapper-Geräte zu definieren, wie in der **12-dm-permissions.rules**-Datei dokumentiert.

### A.3.2. udev unterstützende Befehle und Schnittstellen

Tabelle A.2, „dmsetup-Befehle zur Unterstützung von udev“ fasst die **dmsetup**-Befehle zusammen, die **udev**-Integration unterstützen.

**Tabelle A.2. dmsetup-Befehle zur Unterstützung von udev**

| Befehl | Beschreibung |
|---|---|
| **dmsetup udevcomplete** | Wird verwendet zur Benachrichtigung, dass udev die Verarbeitung der Regeln abgeschlossen hat und entsperrt den wartenden Prozess (aufgerufen von den **udev**-Regeln in **95-dm-notify.rules**). |
| **dmsetup udevcomplete_all** | Wird verwendet zur Suche und Bereinigung von Programmfehlern, um manuell alle wartenden Prozesse zu entsperren. |
| **dmsetup udevcookies** | Wird verwendet zur Suche und Bereinigung von Programmfehlern, um alle vorhandenen Cookies anzuzeigen (systemweite Semaphore). |
| **dmsetup udevcreatecookie** | Wird verwendet zur manuellen Erstellung eines Cookies (Semaphors). Dies ist hilfreich, um mehrere Prozesse unter einer Synchronisationsressource auszuführen. |
| **dmsetup udevreleasecookie** | Wird verwendet, um auf sämtliche **udev**-Verarbeitung im Zusammenhang mit allen Prozessen, die unter das eine Synchronisations-Cookie platziert wurden, zu warten. |

Die **dmsetup**-Optionen, die **udev**-Integration unterstützen, lauten folgendermaßen.

**--udevcookie**
: Muss für alle dmsetup-Prozesse definiert werden, die wir in einer udev-Transaktion hinzufügen möchten. Wird in Verbindung mit **udevcreatecookie** und **udevreleasecookie** verwendet:

```
COOKIE=$(dmsetup udevcreatecookie)
  dmsetup command --udevcookie $COOKIE ....
  dmsetup command --udevcookie $COOKIE ....
  ....
  dmsetup command --udevcookie $COOKIE ....
dmsetup udevreleasecookie --udevcookie $COOKIE
```

Neben der Verwendung der **--udevcookie**-Option können Sie auch einfach die Variable in eine Umgebung des Prozesses exportieren:

```
export DM_UDEV_COOKIE=$(dmsetup udevcreatecookie)
  dmsetup command ...
  dmsetup command ...
  ...
  dmsetup command ...
```

**--noudevrules**

Deaktiviert udev-Regeln. Knoten/Symlinks werden durch **libdevmapper** selbst erzeugt (auf die alte Art). Diese Option steht für die Suche und Bereinigung von Programmfehlern zur Verfügung, wenn **udev** nicht einwandfrei funktioniert.

**--noudevsync**

Deaktiviert **udev**-Synchronisation. Ebenfalls zum Zwecke der Suche und Bereinigung von Programmfehlern.

Werfen Sie einen Blick auf die Handbuchseite (8) von **dmsetup** für Informationen zum Befehl **dmsetup** und dessen Optionen.

Die LVM-Befehle unterstützen die folgenden Optionen, die **udev**-Integration unterstützen:

- **--noudevrules**: wie auch beim **dmsetup**-Befehl deaktiviert dies die **udev**-Regeln.
- **--noudevsync**: wie auch beim **dmsetup**-Befehl deaktiviert dies die **udev**-Synchronisation.

Die **lvm.conf**-Datei umfasst die folgenden Optionen, die **udev**-Integration unterstützen:

- **udev_rules**: aktiviert/deaktiviert **udev_rules** für alle LVM2-Befehle übergreifend
- **udev_sync**: aktiviert/deaktiviert **udev**-Synchronisation für alle LVM-Befehle übergreifend

Weitere Informationen über die **lvm.conf**-Dateioptionen finden Sie in den Kommentaren der **lvm.conf**-Datei.

# Die LVM-Konfigurationsdateien

LVM unterstützt mehrere Konfigurationsdateien. Während des Systemstarts wird die Konfigurationsdatei **`lvm.conf`** aus dem Verzeichnis geladen, das durch die Systemvariable **`LVM_SYSTEM_DIR`** spezifiziert wird und die standardmäßig auf **`/etc/lvm`** gesetzt ist.

In der Datei **`lvm.conf`** können zusätzliche Konfigurationsdateien angegeben werden, die geladen werden sollen. Einstellungen in nachfolgenden Dateien überschreiben Einstellungen von zuvor geladenen Dateien. Um die Einstellungen anzuzeigen, die nach Laden aller Konfigurationsdateien verwendet werden, führen Sie den Befehl **`lvm dumpconfig`** aus.

Für Informationen zum Laden zusätzlicher Konfigurationsdateien werfen Sie einen Blick auf Abschnitt C.2, „Host-Tags“.

## B.1. Die LVM-Konfigurationsdateien

Die folgenden Dateien werden zur Konfiguration von LVM verwendet:

**/etc/lvm/lvm.conf**

Zentrale Konfigurationsdatei, die von den Tools gelesen wird.

**etc/lvm/lvm_*hosttag*.conf**

Für jeden Host-Tag wird eine extra Konfigurationsdatei gelesen, falls vorhanden: **`lvm_`*`hosttag`*`.conf`**. Falls diese Datei neue Tags definiert, dann werden weitere Konfigurationsdateien, die eingelesen werden sollen, an die Liste der Tiles angehängt. Werfen Sie einen Blick auf Abschnitt C.2, „Host-Tags“ für Informationen zu Host-Tags.

Zusätzlich zu den LVM-Konfigurationsdateien umfasst ein System, auf dem LVM läuft, die folgenden Dateien, die Auswirkungen auf die Einstellung des LVM-Systems haben:

**/etc/lvm/.cache**

Gerätenamens-Filter Cache-Datei (konfigurierbar).

**/etc/lvm/backup/**

Verzeichnis für Backups für automatische Datenträgergruppen-Metadaten (konfigurierbar).

**/etc/lvm/archive/**

Verzeichnis für Archive für automatische Datenträgergruppen-Metadaten (konfigurierbar in Bezug auf Verzeichnispfad und Tiefe des Archivverlaufs).

**/var/lock/lvm/**

In einer Einzel-Host-Konfiguration verhindern Lock-Dateien, dass Tools, die parallel laufen, Metadaten korrumpieren. In einem Cluster wird clusterweites DLM verwendet.

## B.2. Beispiel einer lvm.conf-Datei

Nachfolgend finden Sie ein Beispiel für eine **`lvm.conf`**-Konfigurationsdatei. Ihre Konfigurationsdatei

kann sich von der gezeigten leicht unterscheiden.

```
# This is an example configuration file for the LVM2 system.
# It contains the default settings that would be used if there was no
# /etc/lvm/lvm.conf file.
#
# Refer to 'man lvm.conf' for further information including the file layout.
#
# To put this file in a different directory and override /etc/lvm set
# the environment variable LVM_SYSTEM_DIR before running the tools.


# This section allows you to configure which block devices should
# be used by the LVM system.
devices {

    # Where do you want your volume groups to appear ?
    dir = "/dev"

    # An array of directories that contain the device nodes you wish
    # to use with LVM2.
    scan = [ "/dev" ]

    # If several entries in the scanned directories correspond to the
    # same block device and the tools need to display a name for device,
    # all the pathnames are matched against each item in the following
    # list of regular expressions in turn and the first match is used.
    # preferred_names = [ ]

    # Try to avoid using undescriptive /dev/dm-N names, if present.
    preferred_names = [ "^/dev/mpath/", "^/dev/mapper/mpath", "^/dev/[hs]d" ]

    # A filter that tells LVM2 to only use a restricted set of devices.
    # The filter consists of an array of regular expressions.  These
    # expressions can be delimited by a character of your choice, and
    # prefixed with either an 'a' (for accept) or 'r' (for reject).
    # The first expression found to match a device name determines if
    # the device will be accepted or rejected (ignored).  Devices that
    # don't match any patterns are accepted.

    # Be careful if there there are symbolic links or multiple filesystem
    # entries for the same device as each name is checked separately against
    # the list of patterns.  The effect is that if any name matches any 'a'
    # pattern, the device is accepted; otherwise if any name matches any 'r'
    # pattern it is rejected; otherwise it is accepted.

    # Don't have more than one filter line active at once: only one gets used.

    # Run vgscan after you change this parameter to ensure that
    # the cache file gets regenerated (see below).
    # If it doesn't do what you expect, check the output of 'vgscan -vvvv'.


    # By default we accept every block device:
    filter = [ "a/.*/" ]

    # Exclude the cdrom drive
    # filter = [ "r|/dev/cdrom|" ]

    # When testing I like to work with just loopback devices:
    # filter = [ "a/loop/", "r/.*/" ]
```

```
    # Or maybe all loops and ide drives except hdc:
    # filter =[ "a|loop|", "r|/dev/hdc|", "a|/dev/ide|", "r|.*|" ]

    # Use anchors if you want to be really specific
    # filter = [ "a|^/dev/hda8$|", "r/.*/" ]

    # The results of the filtering are cached on disk to avoid
    # rescanning dud devices (which can take a very long time).
    # By default this cache is stored in the /etc/lvm/cache directory
    # in a file called '.cache'.
    # It is safe to delete the contents: the tools regenerate it.
    # (The old setting 'cache' is still respected if neither of
    # these new ones is present.)
    cache_dir = "/etc/lvm/cache"
    cache_file_prefix = ""

    # You can turn off writing this cache file by setting this to 0.
    write_cache_state = 1

    # Advanced settings.

    # List of pairs of additional acceptable block device types found
    # in /proc/devices with maximum (non-zero) number of partitions.
    # types = [ "fd", 16 ]

    # If sysfs is mounted (2.6 kernels) restrict device scanning to
    # the block devices it believes are valid.
    # 1 enables; 0 disables.
    sysfs_scan = 1

    # By default, LVM2 will ignore devices used as components of
    # software RAID (md) devices by looking for md superblocks.
    # 1 enables; 0 disables.
    md_component_detection = 1

    # By default, if a PV is placed directly upon an md device, LVM2
    # will align its data blocks with the md device's stripe-width.
    # 1 enables; 0 disables.
    md_chunk_alignment = 1

    # Default alignment of the start of a data area in MB.  If set to 0,
    # a value of 64KB will be used.  Set to 1 for 1MiB, 2 for 2MiB, etc.
    # default_data_alignment = 1

    # By default, the start of a PV's data area will be a multiple of
    # the 'minimum_io_size' or 'optimal_io_size' exposed in sysfs.
    # - minimum_io_size - the smallest request the device can perform
    #   w/o incurring a read-modify-write penalty (e.g. MD's chunk size)
    # - optimal_io_size - the device's preferred unit of receiving I/O
    #   (e.g. MD's stripe width)
    # minimum_io_size is used if optimal_io_size is undefined (0).
    # If md_chunk_alignment is enabled, that detects the optimal_io_size.
    # This setting takes precedence over md_chunk_alignment.
    # 1 enables; 0 disables.
    data_alignment_detection = 1

    # Alignment (in KB) of start of data area when creating a new PV.
    # md_chunk_alignment and data_alignment_detection are disabled if set.
    # Set to 0 for the default alignment (see: data_alignment_default)
    # or page size, if larger.
```

```
    data_alignment = 0

    # By default, the start of the PV's aligned data area will be shifted by
    # the 'alignment_offset' exposed in sysfs.  This offset is often 0 but
    # may be non-zero; e.g.: certain 4KB sector drives that compensate for
    # windows partitioning will have an alignment_offset of 3584 bytes
    # (sector 7 is the lowest aligned logical block, the 4KB sectors start
    # at LBA -1, and consequently sector 63 is aligned on a 4KB boundary).
    # But note that pvcreate --dataalignmentoffset will skip this detection.
    # 1 enables; 0 disables.
    data_alignment_offset_detection = 1

    # If, while scanning the system for PVs, LVM2 encounters a device-mapper
    # device that has its I/O suspended, it waits for it to become accessible.
    # Set this to 1 to skip such devices.  This should only be needed
    # in recovery situations.
    ignore_suspended_devices = 0

    # During each LVM operation errors received from each device are counted.
    # If the counter of a particular device exceeds the limit set here, no
    # further I/O is sent to that device for the remainder of the respective
    # operation. Setting the parameter to 0 disables the counters altogether.
    disable_after_error_count = 0

    # Allow use of pvcreate --uuid without requiring --restorefile.
    require_restorefile_with_uuid = 1
}

# This section allows you to configure the way in which LVM selects
# free space for its Logical Volumes.
#allocation {
#    When searching for free space to extend an LV, the "cling"
#    allocation policy will choose space on the same PVs as the last
#    segment of the existing LV.  If there is insufficient space and a
#    list of tags is defined here, it will check whether any of them are
#    attached to the PVs concerned and then seek to match those PV tags
#    between existing extents and new extents.
#    Use the special tag "@*" as a wildcard to match any PV tag.
#
#    Example: LVs are mirrored between two sites within a single VG.
#    PVs are tagged with either @site1 or @site2 to indicate where
#    they are situated.
#
#    cling_tag_list = [ "@site1", "@site2" ]
#    cling_tag_list = [ "@*" ]
#}

# This section that allows you to configure the nature of the
# information that LVM2 reports.
log {

    # Controls the messages sent to stdout or stderr.
    # There are three levels of verbosity, 3 being the most verbose.
    verbose = 0

    # Should we send log messages through syslog?
    # 1 is yes; 0 is no.
    syslog = 1

    # Should we log error and debug messages to a file?
```

```
    # By default there is no log file.
    #file = "/var/log/lvm2.log"

    # Should we overwrite the log file each time the program is run?
    # By default we append.
    overwrite = 0

    # What level of log messages should we send to the log file and/or syslog?
    # There are 6 syslog-like log levels currently in use - 2 to 7 inclusive.
    # 7 is the most verbose (LOG_DEBUG).
    level = 0

    # Format of output messages
    # Whether or not (1 or 0) to indent messages according to their severity
    indent = 1

    # Whether or not (1 or 0) to display the command name on each line output
    command_names = 0

    # A prefix to use before the message text (but after the command name,
    # if selected).  Default is two spaces, so you can see/grep the severity
    # of each message.
    prefix = "  "

    # To make the messages look similar to the original LVM tools use:
    #   indent = 0
    #   command_names = 1
    #   prefix = " -- "

    # Set this if you want log messages during activation.
    # Don't use this in low memory situations (can deadlock).
    # activation = 0
}

# Configuration of metadata backups and archiving.  In LVM2 when we
# talk about a 'backup' we mean making a copy of the metadata for the
# *current* system.  The 'archive' contains old metadata configurations.
# Backups are stored in a human readeable text format.
backup {

    # Should we maintain a backup of the current metadata configuration ?
    # Use 1 for Yes; 0 for No.
    # Think very hard before turning this off!
    backup = 1

    # Where shall we keep it ?
    # Remember to back up this directory regularly!
    backup_dir = "/etc/lvm/backup"

    # Should we maintain an archive of old metadata configurations.
    # Use 1 for Yes; 0 for No.
    # On by default.  Think very hard before turning this off.
    archive = 1

    # Where should archived files go ?
    # Remember to back up this directory regularly!
    archive_dir = "/etc/lvm/archive"

    # What is the minimum number of archive files you wish to keep ?
    retain_min = 10
```

```
    # What is the minimum time you wish to keep an archive file for ?
    retain_days = 30
}

# Settings for the running LVM2 in shell (readline) mode.
shell {

    # Number of lines of history to store in ~/.lvm_history
    history_size = 100
}


# Miscellaneous global LVM2 settings
global {

    # The file creation mask for any files and directories created.
    # Interpreted as octal if the first digit is zero.
    umask = 077

    # Allow other users to read the files
    #umask = 022

    # Enabling test mode means that no changes to the on disk metadata
    # will be made.  Equivalent to having the -t option on every
    # command.  Defaults to off.
    test = 0

    # Default value for --units argument
    units = "h"

    # Since version 2.02.54, the tools distinguish between powers of
    # 1024 bytes (e.g. KiB, MiB, GiB) and powers of 1000 bytes (e.g.
    # KB, MB, GB).
    # If you have scripts that depend on the old behaviour, set this to 0
    # temporarily until you update them.
    si_unit_consistency = 1

    # Whether or not to communicate with the kernel device-mapper.
    # Set to 0 if you want to use the tools to manipulate LVM metadata
    # without activating any logical volumes.
    # If the device-mapper kernel driver is not present in your kernel
    # setting this to 0 should suppress the error messages.
    activation = 1

    # If we can't communicate with device-mapper, should we try running
    # the LVM1 tools?
    # This option only applies to 2.4 kernels and is provided to help you
    # switch between device-mapper kernels and LVM1 kernels.
    # The LVM1 tools need to be installed with .lvm1 suffices
    # e.g. vgscan.lvm1 and they will stop working after you start using
    # the new lvm2 on-disk metadata format.
    # The default value is set when the tools are built.
    # fallback_to_lvm1 = 0

    # The default metadata format that commands should use - "lvm1" or "lvm2".
    # The command line override is -M1 or -M2.
    # Defaults to "lvm2".
    # format = "lvm2"
```

```
    # Location of proc filesystem
    proc = "/proc"

    # Type of locking to use. Defaults to local file-based locking (1).
    # Turn locking off by setting to 0 (dangerous: risks metadata corruption
    # if LVM2 commands get run concurrently).
    # Type 2 uses the external shared library locking_library.
    # Type 3 uses built-in clustered locking.
    # Type 4 uses read-only locking which forbids any operations that might
    # change metadata.
    locking_type = 1

    # Set to 0 to fail when a lock request cannot be satisfied immediately.
    wait_for_locks = 1

    # If using external locking (type 2) and initialisation fails,
    # with this set to 1 an attempt will be made to use the built-in
    # clustered locking.
    # If you are using a customised locking_library you should set this to 0.
    fallback_to_clustered_locking = 1

    # If an attempt to initialise type 2 or type 3 locking failed, perhaps
    # because cluster components such as clvmd are not running, with this set
    # to 1 an attempt will be made to use local file-based locking (type 1).
    # If this succeeds, only commands against local volume groups will proceed.
    # Volume Groups marked as clustered will be ignored.
    fallback_to_local_locking = 1

    # Local non-LV directory that holds file-based locks while commands are
    # in progress.  A directory like /tmp that may get wiped on reboot is OK.
    locking_dir = "/var/lock/lvm"

    # Whenever there are competing read-only and read-write access requests for
    # a volume group's metadata, instead of always granting the read-only
    # requests immediately, delay them to allow the read-write requests to be
    # serviced.  Without this setting, write access may be stalled by a high
    # volume of read-only requests.
    # NB. This option only affects locking_type = 1 viz. local file-based
    # locking.
    prioritise_write_locks = 1

    # Other entries can go here to allow you to load shared libraries
    # e.g. if support for LVM1 metadata was compiled as a shared library use
    #   format_libraries = "liblvm2format1.so"
    # Full pathnames can be given.

    # Search this directory first for shared libraries.
    #   library_dir = "/lib"

    # The external locking library to load if locking_type is set to 2.
    #   locking_library = "liblvm2clusterlock.so"

    # Treat any internal errors as fatal errors, aborting the process that
    # encountered the internal error. Please only enable for debugging.
    abort_on_internal_errors = 0

    # If set to 1, no operations that change on-disk metadata will be permitted.
    # Additionally, read-only commands that encounter metadata in need of repair
    # will still be allowed to proceed exactly as if the repair had been
    # performed (except for the unchanged vg_seqno).
```

```
    # Inappropriate use could mess up your system, so seek advice first!
    metadata_read_only = 0
}

activation {
    # Set to 0 to disable udev synchronisation (if compiled into the binaries).
    # Processes will not wait for notification from udev.
    # They will continue irrespective of any possible udev processing
    # in the background.  You should only use this if udev is not running
    # or has rules that ignore the devices LVM2 creates.
    # The command line argument --nodevsync takes precedence over this setting.
    # If set to 1 when udev is not running, and there are LVM2 processes
    # waiting for udev, run 'dmsetup udevcomplete_all' manually to wake them up.
    udev_sync = 1

    # Set to 0 to disable the udev rules installed by LVM2 (if built with
    # --enable-udev_rules). LVM2 will then manage the /dev nodes and symlinks
    # for active logical volumes directly itself.
    # N.B. Manual intervention may be required if this setting is changed
    # while any logical volumes are active.
    udev_rules = 1

    # How to fill in missing stripes if activating an incomplete volume.
    # Using "error" will make inaccessible parts of the device return
    # I/O errors on access.  You can instead use a device path, in which
    # case, that device will be used to in place of missing stripes.
    # But note that using anything other than "error" with mirrored
    # or snapshotted volumes is likely to result in data corruption.
    missing_stripe_filler = "error"

    # How much stack (in KB) to reserve for use while devices suspended
    reserved_stack = 256

    # How much memory (in KB) to reserve for use while devices suspended
    reserved_memory = 8192

    # Nice value used while devices suspended
    process_priority = -18

    # If volume_list is defined, each LV is only activated if there is a
    # match against the list.
    #   "vgname" and "vgname/lvname" are matched exactly.
    #   "@tag" matches any tag set in the LV or VG.
    #   "@*" matches if any tag defined on the host is also set in the LV or VG
    #
    # volume_list = [ "vg1", "vg2/lvol1", "@tag1", "@*" ]

    # Size (in KB) of each copy operation when mirroring
    mirror_region_size = 512

    # Setting to use when there is no readahead value stored in the metadata.
    #
    # "none" - Disable readahead.
    # "auto" - Use default value chosen by kernel.
    readahead = "auto"

    # 'mirror_image_fault_policy' and 'mirror_log_fault_policy' define
    # how a device failure affecting a mirror is handled.
    # A mirror is composed of mirror images (copies) and a log.
    # A disk log ensures that a mirror does not need to be re-synced
```

```
    # (all copies made the same) every time a machine reboots or crashes.
    #
    # In the event of a failure, the specified policy will be used to determine
    # what happens. This applies to automatic repairs (when the mirror is being
    # monitored by dmeventd) and to manual lvconvert --repair when
    # --use-policies is given.
    #
    # "remove" - Simply remove the faulty device and run without it.  If
    #            the log device fails, the mirror would convert to using
    #            an in-memory log.  This means the mirror will not
    #            remember its sync status across crashes/reboots and
    #            the entire mirror will be re-synced.  If a
    #            mirror image fails, the mirror will convert to a
    #            non-mirrored device if there is only one remaining good
    #            copy.
    #
    # "allocate" - Remove the faulty device and try to allocate space on
    #            a new device to be a replacement for the failed device.
    #            Using this policy for the log is fast and maintains the
    #            ability to remember sync state through crashes/reboots.
    #            Using this policy for a mirror device is slow, as it
    #            requires the mirror to resynchronize the devices, but it
    #            will preserve the mirror characteristic of the device.
    #            This policy acts like "remove" if no suitable device and
    #            space can be allocated for the replacement.
    #
    # "allocate_anywhere" - Not yet implemented. Useful to place the log device
    #            temporarily on same physical volume as one of the mirror
    #            images. This policy is not recommended for mirror devices
    #            since it would break the redundant nature of the mirror. This
    #            policy acts like "remove" if no suitable device and space can
    #            be allocated for the replacement.

    mirror_log_fault_policy = "allocate"
    mirror_image_fault_policy = "remove"

    # 'snapshot_autoextend_threshold' and 'snapshot_autoextend_percent' define
    # how to handle automatic snapshot extension. The former defines when the
    # snapshot should be extended: when its space usage exceeds this many
    # percent. The latter defines how much extra space should be allocated for
    # the snapshot, in percent of its current size.
    #
    # For example, if you set snapshot_autoextend_threshold to 70 and
    # snapshot_autoextend_percent to 20, whenever a snapshot exceeds 70% usage,
    # it will be extended by another 20%. For a 1G snapshot, using up 700M will
    # trigger a resize to 1.2G. When the usage exceeds 840M, the snapshot will
    # be extended to 1.44G, and so on.
    #
    # Setting snapshot_autoextend_threshold to 100 disables automatic
    # extensions. The minimum value is 50 (A setting below 50 will be treated
    # as 50).

    snapshot_autoextend_threshold = 100
    snapshot_autoextend_percent = 20

    # While activating devices, I/O to devices being (re)configured is
    # suspended, and as a precaution against deadlocks, LVM2 needs to pin
    # any memory it is using so it is not paged out.  Groups of pages that
    # are known not to be accessed during activation need not be pinned
    # into memory.  Each string listed in this setting is compared against
```

```
    # each line in /proc/self/maps, and the pages corresponding to any
    # lines that match are not pinned.  On some systems locale-archive was
    # found to make up over 80% of the memory used by the process.
    # mlock_filter = [ "locale/locale-archive", "gconv/gconv-modules.cache" ]

    # Set to 1 to revert to the default behaviour prior to version 2.02.62
    # which used mlockall() to pin the whole process's memory while activating
    # devices.
    use_mlockall = 0

    # Monitoring is enabled by default when activating logical volumes.
    # Set to 0 to disable monitoring or use the --ignoremonitoring option.
    monitoring = 1

    # When pvmove or lvconvert must wait for the kernel to finish
    # synchronising or merging data, they check and report progress
    # at intervals of this number of seconds.  The default is 15 seconds.
    # If this is set to 0 and there is only one thing to wait for, there
    # are no progress reports, but the process is awoken immediately the
    # operation is complete.
    polling_interval = 15
}


####################
# Advanced section #
####################

# Metadata settings
#
# metadata {
    # Default number of copies of metadata to hold on each PV.  0, 1 or 2.
    # You might want to override it from the command line with 0
    # when running pvcreate on new PVs which are to be added to large VGs.

    # pvmetadatacopies = 1

    # Default number of copies of metadata to maintain for each VG.
    # If set to a non-zero value, LVM automatically chooses which of
    # the available metadata areas to use to achieve the requested
    # number of copies of the VG metadata.  If you set a value larger
    # than the the total number of metadata areas available then
    # metadata is stored in them all.
    # The default value of 0 ("unmanaged") disables this automatic
    # management and allows you to control which metadata areas
    # are used at the individual PV level using 'pvchange
    # --metadataignore y/n'.

    # vgmetadatacopies = 0

    # Approximate default size of on-disk metadata areas in sectors.
    # You should increase this if you have large volume groups or
    # you want to retain a large on-disk history of your metadata changes.

    # pvmetadatasize = 255

    # List of directories holding live copies of text format metadata.
    # These directories must not be on logical volumes!
    # It's possible to use LVM2 with a couple of directories here,
    # preferably on different (non-LV) filesystems, and with no other
```

```
    # on-disk metadata (pvmetadatacopies = 0). Or this can be in
    # addition to on-disk metadata areas.
    # The feature was originally added to simplify testing and is not
    # supported under low memory situations - the machine could lock up.
    #
    # Never edit any files in these directories by hand unless you
    # you are absolutely sure you know what you are doing! Use
    # the supplied toolset to make changes (e.g. vgcfgrestore).

    # dirs = [ "/etc/lvm/metadata", "/mnt/disk2/lvm/metadata2" ]
#}

# Event daemon
#
dmeventd {
    # mirror_library is the library used when monitoring a mirror device.
    #
    # "libdevmapper-event-lvm2mirror.so" attempts to recover from
    # failures.  It removes failed devices from a volume group and
    # reconfigures a mirror as necessary. If no mirror library is
    # provided, mirrors are not monitored through dmeventd.

    mirror_library = "libdevmapper-event-lvm2mirror.so"

    # snapshot_library is the library used when monitoring a snapshot device.
    #
    # "libdevmapper-event-lvm2snapshot.so" monitors the filling of
    # snapshots and emits a warning through syslog when the use of
    # the snapshot exceeds 80%. The warning is repeated when 85%, 90% and
    # 95% of the snapshot is filled.

    snapshot_library = "libdevmapper-event-lvm2snapshot.so"

    # Full path of the dmeventd binary.
    #
    # executable = "/sbin/dmeventd"
}
```

# LVM Objekt-Tags

Ein LVM-Tag ist ein Wort, das zur Gruppierung von LVM2-Objekten des gleichen Typs verwendet werden kann. Tags können an Objekte wie physische Datenträger, Datenträgergruppen und logische Datenträger angehängt werden. Sie können weiterhin an Hosts in einer Cluster-Konfiguration angehängt werden. Snapshots können nicht mit Tags versehen werden.

Tags können in der Befehlszeile an Stelle von PV, VG oder LV Parametern angegeben werden. Sie sollten mit einem Präfix "@" versehen werden, um eine Mehrdeutigkeit zu vermeiden. Jeder Tag wird erweitert, indem er durch alle Objekte ersetzt wird, die diesen Tag besitzen und die den von der Position auf der Kommandozeile erwarteten Typ besitzen.

Ab der Red Hat Enterprise Linux 6.1 Release dürfen LVM-Tags bis zu 1024 Zeichen lang sein (bei früheren Releases lag die Obergrenze bei 128 Zeichen). LVM-Tags dürfen nicht mit einem Bindestrich beginnen.

Ein gültiger Tag darf nur bestimmte Zeichen enthalten. Für die Red Hat Enterprise Linux 6.0 Release sind lediglich die Zeichen [A-Za-z0-9_+.-] zulässig. Ab der Red Hat Enterprise Linux 6.1 Release wurde die Liste der zulässigen Zeichen erweitert, so dass Tags nun auch die Zeichen "/", "=", "!", ":", "#" und "&" enthalten dürfen.

Lediglich Objekte in einer Datenträgergruppe können mit Tags versehen werden. Physische Datenträger verlieren ihre Tags, wenn sie aus einer Datenträgergruppe entfernt werden. Dies liegt daran, dass Tags als Teil der Metadaten der Datenträgergruppe gespeichert werden und beim Entfernen eines physischen Datenträgers gelöscht werden. Snapshots können nicht mit Tags versehen werden.

Der folgende Befehl listet alle logischen Datenträger mit dem **`database`**-Tag auf.

```
lvs @database
```

## C.1. Hinzufügen und Entfernen von Objekt-Tags

Um Tags zu physischen Datenträgern hinzuzufügen oder von diesen zu entfernen, verwenden Sie die Optionen **`--addtag`** oder **`--deltag`** des Befehls **`pvchange`**.

Um Tags zu einer Datenträgergruppe hinzuzufügen, bzw. von dieser zu entfernen, verwenden Sie die Optionen **`--addtag`** oder **`--deltag`** der Befehle **`vgchange`** oder **`vgcreate`**.

Um Tags zu logischen Datenträgern hinzuzufügen oder von diesen zu entfernen, verwenden Sie die Optionen **`--addtag`** oder **`--deltag`** der Befehle **`lvchange`** oder **`lvcreate`**.

Ab der Red Hat Enterprise Linux 6.1 Release können Sie nun mehrere **`--addtag`** und **`--deltag`**-Parameter innerhalb eines einzigen **`pvchange`**-, **`vgchange`**- oder **`lvchange`**-Befehls spezifizieren. Beispielsweise löscht der folgende Befehl die Tags **`T9`** und **`T10`** und fügt die Tags **`T13`** und **`T14`** zur Datenträgergruppe **`grant`** hinzu.

```
vgchange --deltag T9 --deltag T10 --addtag T13 --addtag T14 grant
```

## C.2. Host-Tags

In einer Cluster-Konfiguration können Sie Host-Tags in den Konfigurationsdateien definieren. Wenn Sie **`hosttags = 1`** im Abschnitt **`tags`** setzen, wird automatisch ein Host-Tag definiert, der den Hostnamen der Maschine verwendet. Auf diese Weise können Sie eine allgemeine Konfigurationsdatei verwenden,

die auf allen Ihren Maschinen repliziert werden kann, so dass diese identische Kopien dieser Datei besitzen, sich das Verhalten jedoch bezüglich des Hostnamens zwischen den Maschinen unterscheiden kann.

Werfen Sie einen Blick auf Anhang B, *Die LVM-Konfigurationsdateien* für Informationen zu den Konfigurationsdateien.

Für jeden Host-Tag wird eine extra Konfigurationsdatei gelesen, falls sie existiert: lvm_***hosttag.***conf. Falls diese Datei neue Tags definiert, dann werden weitere Konfigurationsdateien an die Liste der einzulesenden Dateien angehängt.

Der folgende Eintrag in der Konfigurationsdatei definiert beispielsweise immer **tag1**, und definiert **tag2**, falls der Hostname **host1** ist.

```
tags { tag1 { }  tag2 { host_list = ["host1"] } }
```

## C.3. Aktivierung mit Tags kontrollieren

Sie können in der Konfigurationsdatei angeben, dass nur bestimmte logische Datenträger auf diesem Host aktiviert werden sollen. Der folgende Eintrag agiert beispielsweise als ein Filter für Aktivierungsanfragen (wie z.B. **vgchange -ay**) und aktiviert lediglich **vg1/lvol0** und jeden logischen Datenträger oder Datenträgergruppe mit dem **database**-Tag in den Metadaten auf diesem Host.

```
activation { volume_list = ["vg1/lvol0", "@database" ] }
```

Es gibt ein spezielles Suchmuster "@*", das nur dann zu einem Treffer führt, wenn irgendein Metadaten-Tag mit irgendeinem Host-Tag auf diese Maschine übereinstimmt.

Stellen Sie sich als weiteres Beispiel eine Situation vor, in der jede Maschine im Cluster den folgenden Eintrag in der Konfigurationsdatei besitzt:

```
tags { hosttags = 1 }
```

Falls Sie **vg1/lvol2** nur auf Host **db2** aktivieren möchten, tun Sie Folgendes:

1. Führen Sie den Befehl **lvchange --addtag @db2 vg1/lvol2** von einem beliebigen Host im Cluster aus.
2. Führen Sie den Befehl **lvchange -ay vg1/lvol2** aus.

Diese Lösung hat zur Folge, dass Hostnamen innerhalb der Metadaten der Datenträgergruppe gespeichert werden.

# Metadaten einer LVM-Datenträgergruppe

Die Konfigurationsdetails einer Datenträgergruppe werden als Metadaten bezeichnet. Standardmäßig wird ein identisches Exemplar der Metadaten in jedem Metadatenbereich auf jedem physischen Datenträger innerhalb der Datenträgergruppe bewahrt. LVM-Metadaten sind klein und können als ASCII gespeichert werden.

Falls eine Datenträgergruppe viele physische Datenträger besitzt, ist das Vorliegen mehrerer Exemplare der Metadaten ineffizient. Mit der Option **`--metadatacopies 0`** des Befehls **`pvcreate`** ist es möglich, einen physischen Datenträger ohne eine einzige Kopie der Metadaten zu erstellen. Sobald Sie die Anzahl der Kopien der Metadaten, die der physische Datenträger enthalten wird, ausgewählt haben, können Sie später nicht mehr ändern. Die Auswahl von "0 Kopien" kann bei Konfigurationsänderungen eine schnellere Aktualisierung bedeuten. Beachten Sie jedoch, dass jede Datenträgergruppe zu jedem Zeitpunkt mindestens einen physischen Datenträger mit einem Metadatenbereich besitzen muss (es sei denn, Sie verwenden die erweiterten Konfigurationseinstellungen, die Ihnen das Speichern von Metadaten der Datenträgergruppe auf einem Dateisystem ermöglicht). Falls Sie beabsichtigen, die Datenträgergruppe in der Zukunft aufzuteilen, benötigt jede Datenträgergruppe mindestens ein Exemplar der Metadaten.

Die zentralen Metadaten werden in ASCII gespeichert. Ein Metadatenbereich ist ein zirkulärer Puffer. Neue Metadaten werden an die alten Metadaten angehängt und dann der Zeiger zum Anfangsbereich aktualisiert.

Sie können die Größe des Metadatenbereichs mit der Option **`--metadatasize`** des Befehls **`pvcreate`** angeben. Die Standardgröße ist zu klein für Datenträgergruppen mit vielen logischen oder physischen Datenträgern.

## D.1. Das Label für physische Datenträger

Standardmäßig platziert der Befehl **`pvcreate`** das Label des physischen Datenträgers innerhalb des zweiten 512-Byte Sektors. Dieses Label kann optional in irgendeinem der ersten vier Sektoren platziert werden, da die LVM-Tools, die das System nach einem Label für einen physischen Datenträger absuchen, die ersten vier Sektoren überprüfen. Das Label des physischen Datenträgers beginnt mit dem String **`LABELONE`**.

Das Label des physischen Datenträgers beinhaltet:

- Die UUID des physischen Datenträgers
- Die Größe der Blockgeräte in Bytes
- NULL-terminierte Liste von Orten der Datenbereiche
- NULL-terminierte Listen von Orten der Datenbereiche

Die Positionen der Metadaten werden in Abstand und Größe (in Bytes) gespeichert. Das Label bietet Platz für rund 15 Positionen, aber die LVM Tools nutzen derzeit drei: einen einzelnen Datenbereich und zwei Metadatenbereiche.

## D.2. Inhalte der Metadaten

Die Metadaten der Datenträgergruppe enthalten:

- Informationen, wann und wo sie erstellt wurde
- Informationen über die Datenträgergruppe

Die Informationen zu der Datenträgergruppe umfassen:

- Name und eine eindeutige ID
- eine Versionsnummer, die sich bei jeder Aktualisierung der Metadaten erhöht
- ggf. zugehörige Eigenschaften: Lese-/Schreib-Zugriff? Kann die Größe verändert werden?
- jegliche administrative Einschränkungen bezüglich der Anzahl der physischen Datenträger und der logischen Datenträger, die sie umfassen kann
- die Extent-Größe (in Einheiten der Sektoren, die als 512-Bytes definiert sind)
- eine ungeordnete Liste physischer Datenträger, aus denen die Datenträgergruppe besteht, jeweils mit:
  - dessen UUID, zur Ermittlung des Blockgeräts, das den Datenträger beinhaltet
  - ggf. zugehörige Eigenschaften, wie z.B. ob der physische Datenträger zuweisbar ist
  - den Anfang des Starts des ersten Extents innerhalb des physischen Datenträgers (in Sektoren)
  - die Anzahl der Extents
- eine ungeordnete Liste logischer Datenträger, jeweils bestehend aus
  - einer geordneten Liste logischer Datenträgersegmente. Die Metadaten beinhalten eine Zuordnung für jedes Segment, angewendet auf eine geordnete Liste physischer Datenträgersegmente oder logischer Datenträgersegmente

## D.3. Beispiel-Metadaten

Nachfolgend ist ein Beispiel für Metadaten einer LVM-Datenträgergruppe für eine Datenträgergruppe mit der Bezeichnung **`myvg`** aufgeführt.

```
# Generated by LVM2: Tue Jan 30 16:28:15 2007

contents = "Text Format Volume Group"
version = 1

description = "Created *before* executing 'lvextend -L+5G /dev/myvg/mylv
/dev/sdc'"

creation_host = "tng3-1"        # Linux tng3-1 2.6.18-8.el5 #1 SMP Fri Jan 26
14:15:21 EST 2007 i686
creation_time = 1170196095      # Tue Jan 30 16:28:15 2007

myvg {
        id = "0zd3UT-wbYT-lDHq-lMPs-EjoE-0o18-wL28X4"
        seqno = 3
        status = ["RESIZEABLE", "READ", "WRITE"]
        extent_size = 8192              # 4 Megabytes
        max_lv = 0
        max_pv = 0

        physical_volumes {

                pv0 {
                        id = "ZBW5qW-dXF2-0bGw-ZCad-2RlV-phwu-1c1RFt"
                        device = "/dev/sda"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }

                pv1 {
                        id = "ZHEZJW-MR64-D3QM-Rv7V-Hxsa-zU24-wztY19"
                        device = "/dev/sdb"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }

                pv2 {
                        id = "wCoG4p-55Ui-9tbp-VTEA-jO6s-RAVx-UREW0G"
                        device = "/dev/sdc"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
                }

                pv3 {
                        id = "hGlUwi-zsBg-39FF-do88-pHxY-8XA2-9WKIiA"
                        device = "/dev/sdd"     # Hint only

                        status = ["ALLOCATABLE"]
                        dev_size = 35964301     # 17.1491 Gigabytes
                        pe_start = 384
                        pe_count = 4390 # 17.1484 Gigabytes
```

```
                }
        }
        logical_volumes {

                mylv {
                        id = "GhUYSF-qVM3-rzQo-a6D2-o0aV-LQet-Ur9OF9"
                        status = ["READ", "WRITE", "VISIBLE"]
                        segment_count = 2

                        segment1 {
                                start_extent = 0
                                extent_count = 1280     # 5 Gigabytes

                                type = "striped"
                                stripe_count = 1        # linear

                                stripes = [
                                        "pv0", 0
                                ]
                        }
                        segment2 {
                                start_extent = 1280
                                extent_count = 1280     # 5 Gigabytes

                                type = "striped"
                                stripe_count = 1        # linear

                                stripes = [
                                        "pv1", 0
                                ]
                        }
                }
        }
}
```

# Versionsgeschichte

| **Version 1-5.400** | **2013-10-31** | **Rüdiger Landmann** |
|---|---|---|

Rebuild with publican 4.0.0

| **Version 1-5** | **2012-07-18** | **Anthony Towns** |
|---|---|---|

Rebuild for Publican 3.0

| **Version 2.0-1** | **Thu May 19 2011** | **Steven Levine** |
|---|---|---|

Erste Release für Red Hat Enterprise Linux 6.1

Behebt: #694619
Dokumentiert die neue **`cling`**-Zuweisungsrichtlinie beim Erweitern eines logischen Datenträgers.

Behebt: #682649
Fügt eine Warnung hinzu hinsichtlich dem Ausführen mehrerer Befehle nacheinander zur Mirror-Erstellung auf geclusterten Datenträgern.

Behebt: #674100
Fügt Beispielausgabe für den **`dmsetup ls --tree`**-Befehl hinzu.

Behebt: #694607
Dokumentiert die Unterstützung für das Angeben mehrerer --addtag und --deltag Parameter mit einem einzigen Befehl auf der Befehlszeile.

Behebt: #694604
Dokumentiert die Unterstützung für die erweiterte Zeichenliste in Tags.

Behebt: #694611
Dokumentiert die Unterstützung für Striped-Mirrors.

Behebt: #694616
Dokumentiert die Unterstützung für Snapshots gespiegelter Datenträger.

Behebt: #694618
Dokumentiert die Unterstützung für Snapshots von exklusiv aktivierten Datenträgern.

Behebt: #682648
Dokumentiert, dass ein Mirror-Standbein verschoben werden kann, wenn das Mirror-Standbein neu zugewiesen wurde.

Behebt: #661530
Aktualisiert die Beispiel-**`cluster.conf`**-Datei, um aktuelle Features widerzuspiegeln.

Behebt: #642400
Fügt eine Anmerkung über die Cluster-Protokollverwaltung durch den Cluster-Knoten mit der niedrigsten Cluster-ID hinzu.

Behebt: #663462
Entfernt veraltete Hinweise auf den Xen Virtual Machine Monitor.

| Version 1.0-1 | Wed Nov 10 2010 | Steven Levine |
|---|---|---|

Erste Release für Red Hat Enterprise Linux 6

# Stichwortverzeichnis

## Symbole

## C



## D

## Z